

PERSONAL COMPUTER

CLUB

MENSILE DI PERSONAL E HOME COMPUTER



30

Programmi

per il tuo

Commodore

Supplemento al N. 19 di Personal Computer
Lit. 3.500.

MEE OBIETTIVO HIGH PRECISION



High precision Data Memories
 è tecnologia avanzata di costruzione.
 È il supporto magnetico testato ai limiti
 della resistenza con garanzia di assoluta
 affidabilità.
 È avanguardia tecnologica per assicurare
 la massima protezione dei dati,
 anche, nelle situazioni più critiche.

HIGH PRECISION A COLPO SICURO!



MEE - Memorie per Elaboratori Elettronici S.p.A.
 Forniture per Centri Elaborazione Dati
 Sede Amm.va: 20144 Milano - Via Boni, 29
 Tel. 4988541 (4 linee r.a.) - Telex 324425 MEE-I
 Filiali e Agenzie: Milano - Bergamo - Tonno
 Biella - Padova - Parma - Bologna - Firenze - Ancona
 Roma - Napoli - Catania - Oristano - Bari - Genova
 Bolzano - Mestre



**PERSONAL COMPUTER CLUB
MENSILE DI**

PERSONAL COMPUTING

Anno II - N. 19 - dicembre 1984

Sped. Abb. Post. Gr. III (70%)

Reg. Trib. Milano: n. 262 del 4/5/1983

Una copia Lit. 3.000

arretrati il doppio da richiedere alla
Casa Editrice con pagamento anticipato

SOCIETÀ EDITRICE:

ALFA LINEA

Direzione, redazione

Piazza Cavour, 2 - 20121 Milano

Telefono: (02) 782661/2/3

DIRETTORE RESPONSABILE

Marco Bindi

Redazione

Paolo Tampieri

Marinella Zetti

Collaboratori

Giacomo Buico

Leonardo Felician

Matteo Ferrari

Pier Luigi Maggi

Angelo Magistri

Roberto Marconcini

Roberto Mariani

Gianroberto Negr

Walter Pistarini

Edoardo Piva

Ernesto Sagramoso

Giordano Serafin

DIREZIONE COMMERCIALE

Via Anfiteatro, 15 - 20121 Milano

Tel. (02) 802388/876622/866220

Direttore Commerciale

Gianluca Rivoli

Direzione Pubblicità

Via Anfiteatro, 15 - 20121 Milano

Tel. (02) 802388/876622/866220

Daniela Morandi (Responsabile)

Margherita Del Monaco

Ketty Cusin

Emanuela Manni

Tiziana Belotti (segreteria)

Conc. escl. Pubblicità

per Roma e Lazio

ALFA MEDIA

Via L. Signorelli, 11

00196 Roma - Tel.: (06) 3963942

Marcella Casagni

M. Enrica Castelletti

Diffusione-Abbonamenti

Via Anfiteatro, 15 - 20121 Milano

Tel. (02) 8059425

Vanda Zaglio

AMMINISTRAZIONE

Via Anfiteatro, 15 - 20121 Milano

Tel.: (02) 8059425

Milena Collica

Fotocomposizione: Proget (Milano)

Stampa: Rotolito Lombarda S.p.A. (Milano)

Distribuzione per l'Italia: Messagerie Periodici

via G. Carcano 32 - Milano

Sole agent for distribution abroad: A.I.E. - Agenzia

Italiana di Esportazione S.p.A. - Via Gadames, 89

20151 Milano - Telefono 30.12.200 - Telex 315367.

Tutti i diritti riservati. La riproduzione totale o parziale dei testi è consentita soltanto con l'autorizzazione scritta della casa editrice. Manoscritti e fotografie, anche se non pubblicati, non si restituiscono.

30 programmi per il tuo Commodore

Supplemento al N. 19 di Personal Computer - Lit. 3.500

SOMMARIO

Scrivetevi le vostre avventure	4
Monitor in linguaggio macchina	7
Sprite Editor per gestire la grafica	10
Roll up e roll down per VIC 20	14
Il «Simon» sul C64	15
«Black Box» sul VIC 20	17
Strategia per scacchiera con «Othello» sul VIC 20	18
Il controllo cursore	21
Grafica ad alta risoluzione col VIC 20	22
Realizzare nuovi caratteri	28
Simulare l'istruzione PLOT sul C64	29
Push Over per VIC 20	
Utility e linguaggio macchina per C64	38
Una «vita» per giocare	40
Un Renumber per il VIC 20	41
Allunaggio con il C64	43
Sprite Editor grafico per il C64	
Leggere i diversi floppy	44
A caccia di stelle sul VIC 20	45
Sporting col metodo «Bubble-Sort»	
Problemi di listato	
Crittografare i programmi	46
L'orologio sveglia con il C64	
Il gioco «Isola» per VIC 20 e C64	48
Le prugne per il C64	53
Inserire il titolo sul C64	57
Incorniciare lo schermo	58
Un archivio fonetico	59
I caratteri suonano sul VIC 20	66

Scrivetevi le vostre avventure!

di Walter Pistarini

«**A**dventure» è il nome di un gioco (o di un tipo di gioco) dove vi trovate ad esplorare una fitta rete di stanze, grotte, labirinti, colline o altro, in cerca di tesori senza prezzo. Qualche nome: Adventure, Dungeon

& Dragons, Hobbit, Mad Martha.

Il primo gioco di questo tipo comparve negli anni '70 sui grossi calcolatori. Le sue origini non sono pienamente accertate, ma i più informati tendono a darne la paternità a Willie Crowther, dell'Università di Stanford, che scrisse il programma in Fortran. In seguito questa versione originale è stata riveduta e modificata da Don Woods.

Fino a quel momento, e per lungo tempo, esplorare i meandri di questo gioco richiedeva l'accesso ai grandi calcolatori, visto che occorreva una disponibilità di almeno 200.000 bytes di memoria. Arrivò un bel giorno, eravamo nel 1978, in cui un tipo destinato a diventare famoso si mise in testa di riscrivere il gioco su un minicomputer e più esattamente su un TRS-80 con 16K di

memoria. Il resto è storia. Scott Adams (questo era il suo nome) divenne il famoso (e ricco) autore di numerosi giochi di questo tipo, e fu seguito da frotte di imitatori più o meno bravi. Vi proponiamo un listato-esempio per il Commodore 64.

Come si gioca

Per giocare dovete dare istruzioni al computer sul da farsi, normalmente usando frasi di una o due parole, in inglese. Un comando tipico può essere «Go North», che dice al computer di andare a nord, e via di questo passo. Mano a mano che si trovano oggetti si valuta se è il caso di prenderli o meno (ogni cosa, anche la più insignificante, ha il suo peso), considerando che solitamente non ci si può caricare più di un tanto. Il gioco è particolarmente interessante perché ci sono sempre da scoprire cose nuove, e i suggerimenti o indizi sul cosa fare sono disseminati ovunque. Una cosa è però sapere che questi suggerimenti esistono, altra è estrarli dalle parole che il computer vi dà, capire poi cosa vogliono dire è ancora un'altra cosa. Un altro motivo di successo è dovuto alle risposte che un gioco ben congegnato può fornire: si ha spesso l'impressione di parlare con una persona, e non con una stupida macchina senza cervello.

Veniamo ora alle note dolenti. Esistono numerosissimi giochi di questo genere, per tutti i tipi di Home e Personal Computer, ma sono per la maggior parte in inglese. Esiste anche qualche traduzione, ma niente, per quanto mi risulti al momento, tradotta in «italiano». Questo articolo cercherà di spiegare tutti i trucchi (o almeno i più importanti) che devono essere usati per scrivere questo tipo di gioco. A corredo dell'articolo, come abbiamo già precisato, è presente un listato per C64

	NORD	EST	SUD	OVEST
STANZA DEL MAGO	2	Ø	Ø	Ø
STANZA DEL GUARDIANO	3	Ø	1	4
STANZA DELLA SABBIA	Ø	Ø	2	Ø
CORRIDOIO	2	Ø	Ø	5
STANZA REALE	Ø	4	Ø	Ø

Figura 2

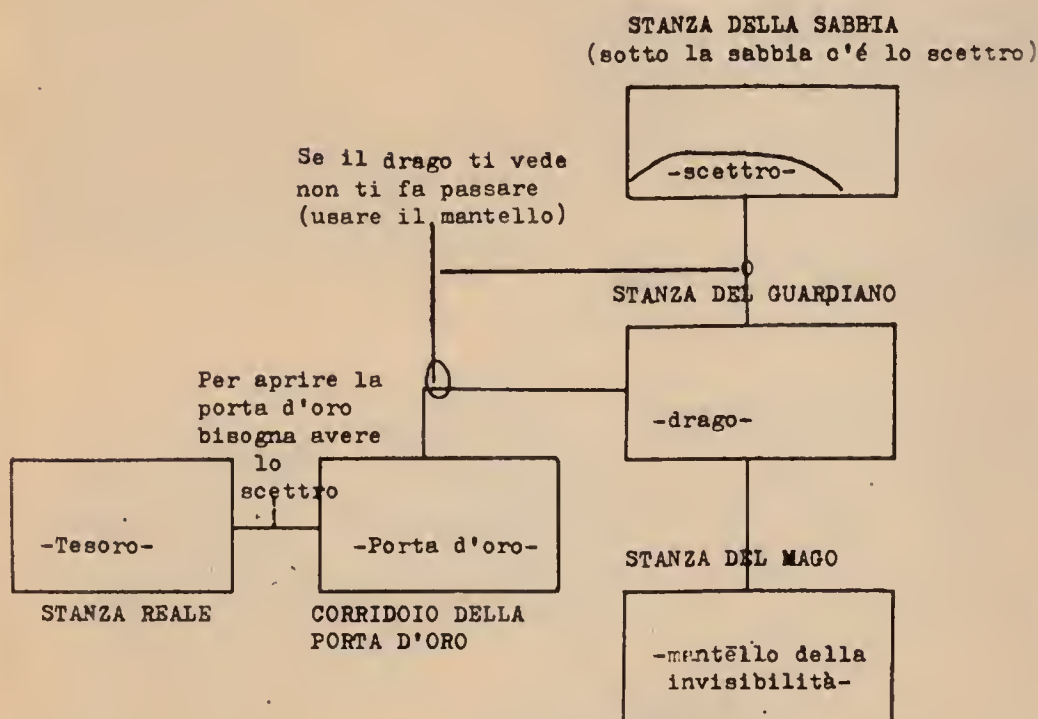


Figura 1

Listato - Avventura

READY.

adattabile facilmente a qualunque computer.

Se vi cimenterete in questo lavoro, scrivendo la «vostra» avventura, vi accorgerete che a fine lavoro avrete incrementato in maniera considerevole la vostra conoscenza del Basic. E questo per una ragione molto semplice: nei programmi di tipo «Adventure» si fa un largo uso di tutte le possibilità del Basic: manipolazione delle stringhe alfanumeriche e delle variabili in maniera molto spinta, generazione dei numeri casuali, logica degli If... THEN, LOOP, presentazione dello schermo, concatenamenti e, se il vostro computer ha una seppur limitata capacità di grafica, colore e suoni, vi potete spingere alla ottimizzazione di semplici effetti sonori e/o grafici. Il tutto si applica ovviamente anche alla programmazione in linguaggio macchina.

Pianificazione dell'avventura

Prima di cominciare a scrivere il programma dovete rispondere alle seguenti domande:

- 1) Dove si svolge l'Avventura?
- 2) Che tipo di mondo si vedrà (reale, fantastico, storico...)?
- 3) Qual è lo scopo principale dell'avventura?
- 4) Che tipo di ostacoli dovranno essere superati e come?

Date queste risposte potete cominciare a fare un disegno della rete di «stanze» (d'ora in avanti userò questo termine per indicare qualunque località, sia essa collina, ruscello, antro o altro) della vostra avventura. Il disegno dovrebbe risultare simile a quello mostrato (solo in piccola parte) in **figura 1**.

Così, per trovare il tesoro bisogna aprire la porta d'oro; per aprire la porta d'oro bisogna avere lo scettro e passare sotto il naso del drago protetti dal mantello dell'invisibilità, mentre per trovare

lo scettro bisogna scavare nella sabbia. Una volta che avete risposto alle domande e disegnato la mappa della vostra avventura siete pronti a cominciare a scrivere il vostro programma.

Il programma

Il programma per gestire questo tipo di gioco è diviso in due parti: il programma vero e proprio che gestisce la logica del tutto e il Data Base, e cioè l'insieme dei dati su cui lavorerà il programma. La parte più importante è appunto il Data Base, in quanto, se il gioco è ben congegnato, cambiando il Data Base si cambierà avventura. Non è facile ottenere questo risultato ma è possibile ed è il sistema usato da molti programmatori di questo tipo di gioco. Vediamo ora separatamente le due parti.

Il data base

Non preoccupatevi del termine tecnico, in pratica dovrete scrivere una lista di lunghezza variabile di istruzioni Data, che conterranno descrizioni, numeri e parole. I modi in cui potete studiare questo insieme di Data sono molti, e qui ne vedremo alcuni, ma l'importante è capire il modo in cui questi dati vengono usati; fatto questo potrete scrivere il tutto a modo vostro.

Il Data Base di ogni avventura, in qualunque modo sia gestito, deve comprendere almeno quattro tabelle (di cui qualcuna «doppia», con una parte numerica e una alfanumerica); il **Vocabolario**, gli **Oggetti**, le **Stanze** e i **Messaggi**. Se si va un po' più nel complesso si può avere anche la tabella dei comandi. Vediamo tutte queste tabelle nel dettaglio.

Vocabolario

Questa tabella (o matrice se preferite) è divisa in due parti: **verbi** e **nomi**. Contiene tutte le parole che l'Avven-

tura può gestire (verbi, oggetti, azioni etc.). I sinonimi (dei verbi) sono gestiti in diversi modi, il più diffuso è quello di far precedere al sinonimo un asterisco (*).

Esempio:

Vai, *cammina, *corri, mangia, *assaggia.

Ovviamente in questo esempio ha assunto che i vari verbi di movimento siano gestiti dal programma tutti alla stessa maniera.

Quando girerà il programma, ci sarà una subroutine che confronterà le due (o una) parole dell'INPUT con questa tabella, e darà come risultato uno o due numeri, identificanti la posizione del verbo e/o del nome nella matrice. Dopodiché la maggior parte del programma giocherà su questi due numeri. Ogni verbo avrà una routine di gestione appropriata, che sarà chiamata con un ON x GOTO, (o, sui personal con la gestione del GOTO a variabili, con un GOTO x). La routine così chiamata farà per prima cosa un controllo sulla «sensatezza» dell'azione (es: MANGIA SEDIA produrrà un bel «Non essere ridicolo» o cose del genere); poi si occuperà della gestione del verbo col nome eventualmente associato.

Oggetti

Questa tabella contiene tutte le informazioni necessarie per gestire i vari oggetti che saranno trovati dal giocatore. È la tabella più modificata nel corso del gioco, perché tutte le volte che il giocatore prenderà o poserà un oggetto, qualcosa viene cambiato in questa matrice. Questa tabella è divisa in due parti: **numero stanza** e **descrizione**. La descrizione contiene appunto la descrizione dell'oggetto. Il numero stanza è invece una tabella numerica contenente il numero della stanza in cui si trova l'oggetto in questo momento. Se il valore della stanza è zero, vuol dire che l'oggetto non è da nessuna

parte (per ora, potrebbe comparire in seguito a certe azioni del giocatore). Se il valore della stanza è meno uno (-1), vuol dire che l'oggetto è in mano al giocatore. Fin qui tutto semplice (spero): quando si vorrà stampare la lista degli oggetti presenti in una stanza, basterà confrontare il numero della stanza in cui si trova il giocatore con il numero contenuto nella parte numerica della tabella degli oggetti. Se il numero è lo stesso, basterà stampare la descrizione dell'oggetto presa dalla corrispondente entrata delle descrizioni della tabella oggetti. Un esempio delle due parti della tabella oggetti potrebbe essere il seguente:

Descrizioni: Ascia, Penna, Bottiglia vuota, Bottiglia piena

Numero stanza: - 1, 2, 5, 0

In questo esempio l'ascia è in mano al giocatore; nella stanza numero cinque c'è una bottiglia vuota, mentre la penna è nella stanza numero due. La bottiglia di acqua piena non è da nessuna parte. Quando il giocatore avrà trovato dell'acqua con cui riempire la bottiglia, il numero di stanza sarà invertito tra bottiglia vuota e bottiglia piena.

Ci sono altri due problemi da risolvere che riguardano gli oggetti: uno è la descrizione stessa, che quando viene data dovrebbe essere lunga (es.: vecchia ascia arrugginita), ma quando il giocatore vuole fare qualcosa con l'oggetto, deve essere breve (es.: prendi ascia, e non prendi vecchia ascia arrugginita).

Vediamo due soluzioni possibili al problema: una è quella di «spaccare» la descrizione in due parti. In questo caso si potranno riempire due elementi della matrice per ogni oggetto (es.: «ascia» e «arrugginita») e stamparli entrambi quando si dà la descrizione dell'oggetto, mentre si può «testare» solo il primo elemento quando si controlla l'INPUT del giocatore. L'altro sistema prevede che sia un

elemento solo a contenere sia la descrizione «lunga» che quella breve, separate da uno slash («/»). In questo caso la tabella descrizioni avrà un solo elemento per ogni oggetto (es: «vecchia ascia arrugginita/ascia/»); lo slash finale serve a capire subito che abbiamo questo tipo di descrizione. Questo sistema l'ho usato nel listato (vedi linee 9390 - 9460 per la lista degli oggetti e linee 56-74 per la stampa degli oggetti senza lo «/»).

L'altro problema che riguarda sempre gli oggetti è il controllo sul fatto se l'oggetto si può veramente prendere o posare (per esempio è meglio evitare che il giocatore possa prendere un albero). Uno dei modi per fare ciò è quello di mettere in una variabile alfanumerica degli «0» se l'oggetto non si può prendere, e degli «1» se l'oggetto relativo si può prendere (se abbiamo dieci oggetti ci sarà una serie di dieci caratteri che saranno o zero o uno a seconda dei casi). Un altro modo è quello di mettere nella tabella comandi (che vedremo più avanti) solo la gestione degli oggetti «prendibili», ma questo sistema è un po' più complesso.

Stanze

La matrice delle stanze contiene tutte le informazioni per ogni «stanza» in cui il giocatore può trovarsi. È anche questa divisa in due tabelle: **Descrizione** e **Direzioni**. La tabella delle descrizioni contiene la descrizione di ogni stanza, senza contenere, molto spesso, l'inizio della frase «Sei in», che sarà invece stampato davanti ad ogni descrizione (questo, ovviamente, per risparmiare memoria). Se poi si volesse, per qualche descrizione, evitare di stampare il «Sei in», si potrebbe mettere un asterisco in fronte alla descrizione, che farà in modo, opportunamente testato, di non stampare la parte di frase succitata. La lista delle

descrizioni comincia alla linea 9010 e la stampa viene fatta alle linee 47-50, testando per l'asterisco.

La tabella delle direzioni è numerica, e contiene una entrata per ogni possibile direzione (per ogni stanza). Le direzioni possono essere quattro, otto o dieci: nord, nord est, est, sud est, sud, sud ovest, ovest, su, giù. Nei listati di esempio sono state considerate solo le quattro direzioni fondamentali. Ogni entrata della tabella consiste nel numero della stanza in cui si troverà il giocatore muovendo in quella direzione. Se c'è uno zero vuol dire che non c'è uscita in quella direzione. Nella **figura due**, se siamo nella stanza numero 2 (la stanza del guardiano) e ci muoviamo a nord, ci troveremo nella stanza della sabbia (stanza numero 3), se invece volessimo andare a sud, ci troveremo nella stanza del mago, e così via. Nel listato a 9300 si trovano le direzioni. Da notare che questa tabella viene usata anche per stampare le uscite visibili, dopo la stampa della descrizione della stanza (80-84 da notare che la stampa delle direzioni viene fatta utilizzando la matrice del vocabolario).

Messaggi

Mettere tutti i messaggi (o almeno la gran parte) che possono essere mostrati al giocatore in una tabella, ha senso specialmente se si usa anche la tabella comandi (vedremo il perché). Comunque nella vostra avventura ci dovranno essere per forza dei messaggi (tipo: «Non succede nulla», oppure «Non puoi!!!» e così via). Nelle avventure più semplici dovendo stampare un messaggio si fa un GOTO ad una linea di programma che stampa il messaggio e ritorna dove deve (il più delle volte ad accettare un altro comando). Questo sistema è stato usato in entrambi i listati (vedere alla linea 3000). Per la gestione dei messaggi

in matrice tenete presente, per ora, che dovete avere una matrice alfabetica contenente tutti i messaggi, e che ovviamente ogni messaggio avrà un numero che lo identifica (il numero di elemento della matrice).

Monitor in linguaggio macchina per VIC20

di Eugenio Rapella

Questo interessante programma, realizzato da **Eugenio Rapella**, è ben fatto e molto utile per la «gestione» di programmi in linguaggio macchina, che stanno prendendo sempre più piede, soprattutto per i VIC-20 che non hanno espansioni di memoria. Caricato il programma Basic, se non si hanno espansioni di memoria, si ha qualcosa come 679 bytes disponibili al linguaggio macchina, che permettono, pur nella loro esiguità, di fare tante belle cosuccie. L'autore si è prodigato molto (e lo vedrete fra poco) nello spiegare come si usa il suo programma con tutti i suoi innumerevoli comandi. Vediamo qualche piccola lacuna nei commenti al listato (i REMarks, per intenderci), che sono presenti, ma non abbondanti né molto dettagliati.

L'idea non è nuova, ma lo sviluppo è molto originale, come pure i comandi che sono a disposizione dell'utente. Detto questo non ci resta che passare la parola al diretto interessato.

La «gestione» (caricamento, esecuzione,...) di un programma in linguaggio macchina (L.M.) per il 6502 attraverso il VIC 20 è, se non si dispone dell'apposita cartidge, piuttosto difficoltosa. Questo «Monitor L.M.», realizzato per un VIC 20 versione standard, può essere un valido aiuto per chi si accinge alle prime esperienze in questo senso. Effettuato il caricamento e l'esecuzione

del programma Basic, viene presentato il «menù» e la «locazione di riferimento in memoria» (il cui valore è stampato in reverse) che, inizialmente, è la 7000 (d'ora in poi il termine «locazione di riferimento» verrà abbreviata con LR).

A questo punto il programma attende:

1) l'inserimento del codice esadecimale della prima istruzione del programma in L.M. che verrà caricata nella LR.

oppure

2) la pressione di uno dei tasti descritti nel «menù» con le relative conseguenze.

L'inserimento di codici differenti da quelli previsti provoca la stampa di un doppio punto interrogativo. Nel primo caso (ad esempio digitando A9), il codice esadecimale viene trasformato in decimale e caricato nella LR; sullo schermo appaiono, sulla stessa riga, i seguenti quattro valori, come in **figura 1**.

Viene quindi stampata la successiva LR in attesa della prossima istruzione. Prima di inserire i codici esadecimali del programma in L.M., è preferibile cancellare il contenuto della memoria ad esso riservato, tramite una delle opzioni disponibili (N=NEW LM).

Vediamo in dettaglio le possibilità offerte dal «menù» (caso 2):

Pressione del tasto:

T = DEC > ESA consente la conversione di un valore decimale in esadecimale.

es. DEC = ? 78

ESA = 4 E

H = ESA > DEC La LR non viene modificata. Consente la conversione da esadecimale a decimale.

es. ESA = ? 1F

DEC = 31

b = LIST PA La LR non viene modificata. La pressione della barra spaziatrice visualizza il contenuto della LR come da **figura 1**. La LR viene incrementata di 1. ↑ = LOC. PRE visualizza il contenuto di LR - 1 (come da **figura 1**) che diviene la

nuova LR.

U = ULTIME visualizza il contenuto delle ultime locazioni di memoria utilizzate nel programma L.M., ovvero quelle il cui contenuto è diverso da zero. Il numero dell'ultima memoria utilizzata è indispensabile per la registrazione di un programma in L.M. su cassetta.

R = RUN L.M. manda in esecuzione il programma L.M. Viene richiesto il valore decimale della locazione di «inizio esecuzione» in modo da consentire l'esecuzione di segmenti del programma L.M. redatto o l'esecuzione di un programma caricato in altra zona. Il valore iniziale normale è 7000. Terminata l'esecuzione del programma L.M., la pressione di un tasto qualsiasi consente il rientro al Basic.

L = LOAD LM consente il caricamento di un programma in L.M. da cassetta (opportunamente registrato tramite l'opzione «S»).

Viene richiesto il valore decimale dell'area di memoria a partire da cui il programma L.M. deve essere caricato: il valore normale è 7000. A caricamento effettuato, viene stampato il nome del programma; la LR viene posta a 7000.

S = SAVE LM consente la registrazione su cassetta di un programma in L.M. presente in memoria dalla locazione 7000 fino all'ultima utilizzata (riscontrabile tramite l'opzione «U») il cui valore viene richiesto. Viene anche richiesto un nome di riferimento (TITOLO?) a cui viene automaticamente posto il suffisso «LM» in reverse.

L = LIST... consente di visualizzare i codici macchina contenuti dalla locazione «x»: LOC IN. (DEC) (ovvero «numero della locazione iniziale espresso in decimale») alla locazione «y»: LOC FIN(DEC) (ovvero «numero della locazione finale espressa in decimale»). Deve essere $7000 \leq x \leq y \leq 7679$; un'errata impostazione dei valori provoca un

Listato - Monitor L.M.

```
1 POKE55,88:POKE56,27:CLR:POKE36879,110:PRINT"U=7000 U=7679 C=F GOT0100
10 H$=""
20 IFDTHENA=INT(D/16):H$=MID$("0123456789ABCDEF",1+D-A*16,1)+H$ D=A GOT020
30 RETURN
32 GETR$:IFR$=""THEN32
33 RETURN
40 D=0:IFH$>"THENFORI=ITOLEN(H$) A=ASC(MID$(H$,1,1))-48:D=D+16+A*(A>9)*7 NEXT
42 RETURN
44 F1=0
46 IFABS(INT(X1))-X1ANDX1>=PTHEM50
48 PRINT"X1'DATO NON VALIDO":F1=1:RETURN
50 IFABS(INT(X2))-X2ANDX2<=UTHEM54
52 PRINT"X2'DATO NON VALIDO":F1=1:RETURN
54 IFX1<X2THENRETURN
56 PRINT"X1'<2' I$":F1=1:RETURN
60 PRINT"X=DEC>ESAIH=ESA>DEC"
62 PRINT"=LIST PAI=LOC.PRE"
64 PRINT"U=ULTIME IR=RUN L.M"
66 PRINT"L=LOAD LMIS=SAVE LM"
68 PRINT"E=LIST...I/=DEL..."
70 PRINT"I=INST...IN=NEW LM"
72 PRINT"*=BASIC IM=MENU"
73 PRINT"+=INIZIO IQ=CM DEC"
74 PRINT"-----"
75 RETURN
82 IFCD<480RCD>700RCD>57ANDCD<65THENPRINT"??" GOT0120
84 RETURN
90 R=PEEK(LC):D=R:GOSUB10:A$=H$ D=LC:GOSUB10
93 IF A$=""THEN A$="00"
94 PRINTA$;TAB(3);RSPC(5-LEN(STR$(R)))LC:TAB(3);H$
96 RETURN
100 PRINT"MONITOR L.M. *"
102 PRINT"*"
104 PRINT"* E.RAPELLA 83 *"GOSUB60
120 PRINTTAB(9)"C
125 GOSUB32
130 V=ASC(R$)
135 IFV=32THENLC=C:GOSUB90:C=C+1:GOT0120
137 IFV=94THENC=C-1:LC=C:GOSUB90:GOT0120
140 IFV<77THEN155
150 GOSUB60:GOT0120
155 IFV<95THEN160
157 C=7000:GOT0120
160 IFV<84THEN190
170 INPUT"DEC=":D
180 GOSUB10:PRINT"ESA="H$:GOT0120
190 IFV<72THEN212
200 INPUT"ESA=":H$
210 GOSUB40:PRINT"DEC="D GOT0120
212 IFV<64THEN220
214 INPUT"C.M. IN DEC":D
216 POKEC,D LC=C:GOSUB90:C=C+1 GOT0120
220 IFV<42THEN240
230 PRINT"RIENTRO AL BASIC":END
240 IFV<78THEN255
242 PRINT"X CANCELO PRG.LM CONFIRMI [S/N]"
245 GOSUB32:IFR$="S"THEN250
248 GOT0120
250 FORX=PTOU:POKEX,0:NEXT:RUN
255 IFV<85THEN320
260 FORH=UTOPSTEP-1
270 IFPEEK(H)<0THEN290
280 NEXT
290 IFH<P+3THENH=P+3
300 FORJ=H-3TOH+3:LC=J:GOSUB90:NEXT:GOT0120
320 IFV<82THEN390
330 PRINT"X ESECUZIONE PRG.L.M"
340 PRINT"X SCRIVI LOC D'INIZIO"
350 INPUT"MIN DEC [7000]":E
360 IF E<PORE>UTHEMPRINT"?:GOT0340
380 PRINT"J":SYSE
382 GOSUB32:END
390 IFV<92THEN440
400 INPUT"LOC IN. (DEC)":X
410 INPUT"LOC FIN(DEC)":Y
420 X1=X:X2=Y:GOSUB44:IFF1THEN400
430 FORH=XTOY:LC=H:GOSUB90:NEXT:C=Y+1:GOT0120
440 IFV<47THEN550
450 PRINT"X ELIMINO LM [X,Y] ES.IN"
470 INPUT"LOC X [DEC]":X
480 INPUT"LOC Y [DEC]":Y
490 X1=X:X2=Y:GOSUB44:IFF1THEN450
500 PORT=Y+1TOU
510 POKET=Y+X-1,PEEK(T):NEXT
520 FORT=U-Y+XTOU:POKET,0:NEXT
540 FORT=X-1TOX+1:LC=T:GOSUB90:NEXT:C=X:GOT0120
550 IFV<73THEN670
560 PRINT"X CREA SPAZIO PER I'INS"
570 PRINT"A PARTIRE DA LOC 'M+1'"
590 INPUT"LOC [M]":N
600 INPUT"QUANTI INST":I
```


Listato - Monitor L.M.

```

610 IF I=0 THEN I=20
620 IF M+1>0 THEN PRINT "IMPOSSIBILE (U) *": GOTO 120
630 FORT=U-ITOM+1 STEP -1
640 POKET+1, PEEK(T) NEXT
650 FORT=M+1TOM+1 POKET, 0 NEXT
660 FORT=MTOM+1+2: LC=T: GOSUB 90 NEXT: C=M+1: GOTO 120
670 IF V<83 THEN 730
680 PRINT "REGISTRAZIONE LM"
690 PRINT "ULTIMA MEMORIA"
692 PRINT "UTILIZZATA NEL PRG. LM"
694 INPUT "DEC": N
696 INPUT "TITOLO": T$
697 T$=T$+"LM"
700 OPEN 1, 1, 2, T$
705 PRINT #1, T$
710 FORT=PTOM: PRINT #1, PEEK(T) NEXT
720 CLOSE 1: PRINT "OK, FATTO": C=P: GOTO 120
730 IF V<76 THEN 800
740 PRINT "CARICAMENTO LM"
745 PRINT "DA CASSETTA?"
750 INPUT "1=LOCC [7000]": W
760 OPEN 1, 1, 0, T$
765 INPUT #1, T$
770 INPUT #1, V$: POKET, V$
780 IF ST=64 THEN CLOSE 1: PRINT "CARICATO ": T$: C=P: GOTO 120
790 W=W+1: GOTO 770
800 CD=V: GOSUB 82: PRINT #1
810 GETS$: IF S$="" THEN 810
820 PRINTS$: CD=ASC(S$): GOSUB 82: H$=R$+S$: GOSUB 40
830 POKET, D: LC=C: GOSUB 90: C=C+1: GOTO 120

```

READY.

Commento al listato «Monitor L.M.»

- Istruz. n. 1** Protegge l'area di memoria da 7000 (P) a 7679 (U) dal Basic; determina la colorazione di sfondo e il colore del carattere di stampa.
- 10-30** Subroutine di trasformazione dal decimale D all'esadecimale H\$.
- 32-33** Subroutine d'attesa.
- 40-42** Subroutine di trasformazione dell'esadecimale H\$ nel decimale D.
- 44-56** Subroutine di controllo di validità dati: se $P \leq X1 \leq X2 \leq U$ è $F1 = 0$ (tutto bene), in caso contrario è $F1 = 1$ (errore)
- 60-75** Subroutine: visualizza il menù ovvero l'elenco dei tasti associati alle varie operazioni.
- 82-84** Subroutine: controlla che il tasto premuto sia uno di quelli previsti (codice ESA / tasto «menù»).
- 90-96** Subroutine di stampa: in riferimento, alla locazione LC, calcola e visualizza i valori di fig. 1.
- 100** Inizio del programma principale.
- 120** C'è la locazione di riferimento (LR)
- 135** Gestisce l'opzione «/» (barra spaziatrice)
- 137** Gestisce l'opzione «↑»
- 140-150** Gestisce l'opzione «M»
- 155-157** Gestisce l'opzione «←»
- 160-180** Gestisce l'opzione «T»
- 190-210** Gestisce l'opzione «H»
- 212-216** Gestisce l'opzione « »
- 220-230** Gestisce l'opzione «*»
- 240-250** Gestisce l'opzione «N»
- 255-300** Gestisce l'opzione «U»
- 320-382** Gestisce l'opzione «R»
- 390-430** Gestisce l'opzione «L»
- 440-540** Gestisce l'opzione «/»
- 550-660** Gestisce l'opzione «|»
- (Se non vi è spazio sufficiente per l'inserimento richiesto, viene visualizzato il messaggio: «* IMPOSSIBILE (U) *» con cui si ricorda che, con l'opzione «U», conviene verificare le ultime locazioni utilizzate dal programma L.M.).
- 670-720** Gestisce l'opzione «S»
- 730-790** Gestisce l'opzione «L»
- 800-830** Accetta i codici esadecimali delle istruzioni del programma in linguaggio macchina.

messaggio appropriato. La LR diviene $Y + 1$.

/ = DEL, cancella i codici macchina dalle locazioni «x» a «y» comprese traslando i codici successivi a «y». La «coda finale» viene riempita con valori nulli. Come da esempio in figura 2 alla fine dell'operazione vengono visualizzati i contenuti di «y - 1», «y», «y + 1». La LR viene posta uguale a «y».

I = INST consente l'inserimento di nuove istruzioni esadecimali tra quelle già caricate in memoria creando l'opportuno «spazio» come nell'esempio in figura 3.

Ad operazione avvenuta vengono visualizzati i contenuti delle locazioni in prossimità di quelle da inserire. La LR viene modificata, nell'esempio LR diviene 7002 in attesa delle 3 nuove istruzioni da inserire.

N = NEW LM azzerà i contenuti della memoria riservata al programma L.M. ovvero dalla locazione 7000 alla 7679. Questa operazione va sempre eseguita dopo il caricamento del programma Monitor L.M. quando si intenda redarre un programma L.M. Per evitare pressioni accidentali di questo tasto, viene richiesta una conferma prima di passare all'esecuzione.

* = BASIC termina il programma Monitor L.M. e rientra al Basic.

M = MENU ricorda i tasti associati alle varie operazioni.

← = INIZIO porta la LR al valore 7000.

CM DEC consente l'inserimento del codice di una istruzione L.M. in decimale anziché esadecimale.

Esempio:

C.M. IN DEC ? 141

Effetto:

8D 141 7001 1B59

Questa operazione è particolarmente utile nel caricamento di un programma L.M. già «decimalizzato» come routine in L.M. di un programma redatto in Basic. L'istruzione n. 1 del programma Basic modifica il

Monitor in L.M.

- | | |
|---|-------------|
| 1) Codice esadecimale dell'istruzione | (es.: A9) |
| 2) Codice decimale dell'istruzione | (es.: 169) |
| 3) Locazione in decimale in cui il codice è stato inserito | (es.: 7000) |
| 4) Locazione, in esadecimale, in cui il codice è stato inserito | (es.: 1B58) |

Figura 1

Monitor in L.M.

Situazione	
codice	
ESA	locazione
A9	7000
FF	7001
8D	7002
40	7003
0	7004
0	7005
impostando:	
LOC IN. (DEC) = 7001	
LOC FIN. (DEC) = 7002	
Effetto	
codice	
ESA	locazione
A9	7000
40	7001
0	7002
0	7003
0	7004
0	7005

Figura 2

Situazione:

A9	7000
FF	7001
8D	7002
40	7003
0	7004

...
LOC (M) ? 7001
QUANTI INSERIMENTI
? 3

Nel rettangolo le locazioni visualizzate.

Effetto

A9	7000
FF	7001
00	7002 ← LR
00	7003
00	7004
8D	7005
40	7006
0	7007

Figura 3

puntatore del «limite della memoria» (locazioni 55 e 56), proteggendo l'area di memoria che va dalla locazione 7000 alla 7679 che ri-

sulta quindi disponibile per la memorizzazione delle istruzioni del programma in L.M. Nella realizzazione di questo programma per un VIC versione standard (3,5 K bytes), si è posto il problema di equilibrare la lunghezza del programma Basic con lo spazio disponibile per il codice macchina. Per evitare un'eccessiva occupazione di memoria del programma Basic, non è stato realizzato un controllo completo dei dati in «input», è opportuno comunque ricordare che qualsiasi segnalazione d'errore (come ILLEGAL QUANTITY ERROR...) non pregiudica il contenuto dell'area riservata al programma L.M., che risulta disponibile rimandando in esecuzione il programma MONITOR L.M. In altri termini, durante la redazione di un programma in L.M., è sempre possibile intervenire con il tasto RESTORE senza perdere i codici L.M. già redatti.

È ovvio come, disponendo di una qualsiasi espansione di memoria, sia possibile, modificando i valori nella prima istruzione, migliorare il programma Basic e/o aumentare lo spazio a disposizione per il programma L.M.

Il seguente programma in L.M. (da «VIC REVEALED» di Nick Hampshire, pag. 37) può essere utilizzato come prova su un VIC 20 versione standard:

(0 = zero)

A9 FF 8D 40 03 A2 FF AD
40 03 9D 00 1E

A9 04 9D 00 96 CE 40 03
CA D0 EF 60

il programma realizza la visualizzazione del set di caratteri del VIC 20.

Come descritto in precedenza, è possibile redarre un programma L.M. dalla locazione 7000, registrarlo su nastro e ricaricarlo in una diversa area di memoria. Ovviamente tale operazione è eseguibile per programmi che non contengono salti ad indirizzi assoluti del programma stesso.

È anche ovvio che l'esecuzione di un programma in L.M. si può ottenere rientrando al Basic e digitando l'opportuna istruzione SYS (es. SYS 7000).

Sprite Editor per gestire la grafica di Walter Pistarini

I COMMODORE 64 ha una delle migliori capacità grafiche degli Home Computer sul mercato. «battezzata» Sprite.

Che cos'è uno Sprite? Il nome, di per sé, non significa molto, esattamente Spirito. Folletto; ma il concetto è simile a quello della gestione grafica dei missili sugli ATARI.

Ogni Sprite è una entità ad alta risoluzione, costituita da un insieme di 21 righe di 24 punti ciascuna; ed ogni Sprite (fino ad un massimo di 8 contemporaneamente) è gestito dall'integrato che si cura di tutta la gestione video, chiamato VIC II (Video Interface Chip II). Per programmarne uno, tutto quello che dovete fare è di definire la sua sequenza di bytes, selezionare il suo colore, scegliere la sua posizione (su un asse orizzontale X che va da 0 a 512 e su un asse Y che

va da 0 a 256) ed infine «accenderlo». Cambiando i valori di X ed Y, potete muovere lo Sprite in qualunque posizione dello schermo.

Ma veniamo ora a qualche dettaglio: come ho già detto, fino a otto Sprites possono essere mostrati sullo schermo contemporaneamente.

Ogni Sprite ha un puntatore (lungo 1 byte) sopra il blocco della memoria video. Il puntatore indica un blocco di 64 bytes entro un «banco» di 16K selezionato dal VIC II (voglio dire che il VIC II «vede» solo 16K, e in questi 16K ci devono stare la memoria video, i puntatori agli Sprites e i dati necessari per costruire gli Sprites).

L'ultimo byte dei 64 usati per descrivere uno Sprite è un byte di controllo; gli altri 63 contengono delle informazioni sui «punti» (o «Pixels») che costituiscono lo Sprite. Ogni gruppo di tre bytes rappresenta una riga di 24 punti dello Sprite. Nel modo standard, un bit (ce ne sono otto in un byte) messo a 1 mostra un punto del colore selezionato, mentre un byte messo a 0 mostra cosa c'è sotto di esso, (normalmente lo sfondo, ma potrebbe essere una parte di uno Sprite di più bassa priorità). Associato con ogni Sprite ci sono alcune altre locazioni di memoria nel chip del VIC II. Il registro di abilitazione della visualizzazione ha un bit per ogni Sprite, come pure il registro di abilitazione al multicolore, quello di raddoppio della dimensione orizzontale e quello per la dimensione verticale: c'è poi quello della priorità dello Sprite rispetto allo sfondo, quello che si accorge della collisione fra due Sprites, e quello della collisione dello Sprite con lo sfondo.

Ancora, c'è un byte per ogni posizione verticale dello Sprite, ed un byte per il posizionamento orizzontale. Dato che ci sono più di 256 posizioni orizzontali, c'è anche un byte contenente un bit per ogni Sprite e che serve

Remarks Listato - Sprite Editor

10 Mette lo «Sfondo 0» in blu, dimensiona le matrici e inizializza le variabili X, Y, R, C, S ed SI.

20 Inizializza il VIC II mettendo a zero i vari registri di visualizzazione e raddoppio dello SPRITE, fa un RESTORE (questo perché potremmo arrivare qui anche per resettare il tutto, vedi riga 470, quando si vuole un nuovo schermo) e legge, nella matrice A\$ i nomi di tutti i colori disponibili che saranno poi stampati sulla parte sinistra dello schermo.

30 Riempie tutta la matrice A con dei punti (valore ASCII del punto = 46), questo per riempire la griglia che sarà poi mostrata sullo schermo. Dopodiché riempie tutta la matrice B di zeri, perché, ovviamente, i DATA per costruire in seguito lo SPRITE a questo punto devono essere tutti a zero.

40 Posiziona lo SPRITE N. 2 sullo schermo, con coordinata X=60 e Y=200. Mette a posto il puntatore alle informazioni per costruire lo SPRITE N. 2 (puntatore che si trova a 2042) e gli dice di leggere i 63 bytes necessari all'indirizzo 832. Mette a posto i colori nei registri appropriati, utilizzando i colori che assume alla partenza (nero, blu chiaro, bianco) e che possono essere cambiati in qualunque momento con i tasti F3, F5 ed F7. Da notare che questa riga di programma (come molte altre) serve solo a poter mostrare lo SPRITE nella parte in basso a sinistra dello schermo.

50 Riempie con i dati dello SPRITE (che si trovano nella matrice B), le locazioni di memoria usate dal VIC II per costruire lo SPRITE. Normalmente a questo punto le informazioni sono tutte a zero e non si vedrà nulla, ma se noi leggessimo da cassetta uno SPRITE precedentemente costruito, arrivando qui la matrice B conterrebbe valori «reali». Il secondo POKE serve a visualizzare lo SPRITE ed il terzo a dire al VIC II che lo SPRITE è di tipo multicolore.

60-125 Stampa la lista dei comandi disponibili.

130 Disegna la griglia di punti sullo schermo. Questo viene fatto stampando tutta la matrice A nella parte destra dello schermo.

140 Dopo aver messo X e Y a 1 va a 290 per disegnare il cursore.

150 Legge il comando battuto.

160 Mette il valore corrente letto nella matrice A dentro la griglia

170-185 Spostamenti del cursore: l'unica cosa che fa è di incrementare o decrementare X o Y, X è incrementato o decrementato di 2 perché si lavora su uno SPRITE multicolore. Se X o Y vanno oltre il bordo della griglia, si fa comparire il cursore sul lato opposto.

190 Se si è battuta la freccia verso sinistra, le corrispondenti 2 posizioni orizzontali sono messe al ?

195 Se si è battuto uno dei tre colori possibili, in R viene messo il corrispettivo ASCII del numero battuto (49 per un 1, 50 per un 2, e 51 per un 3) ed il valore di R viene messo nelle due posizioni orizzontali corrispondenti nella matrice A.

200 La gestione del calcolo dello SPRITE viene fatta a partire dalla riga 300

210 Questo comando, come il seguente, è bivalente: Se lo SPRITE è stato raddoppiato su scala X, viene riportato alle dimensioni normali: se è di dimensioni normali viene raddoppiato. In questa riga c'è un esempio significativo di istruzione ABS. Vediamo cosa succede. Se lo SPRITE è stato raddoppiato, in V+29 troveremo un 4 (per lo SPRITE N. 2) e 4-4 fa 0, per cui mettendo 0 in V+29 riporteremo lo SPRITE alle dimensioni normali. Se invece era già di dimensioni normali, con il PEEK (V+29) troveremo 0 e 0-4 fa -4, ma l'istruzione ABS ci dà il valore assoluto di un numero e senza segno, per cui ci dà 4». Facendo il POKE di 4 in V+29 raddoppieremo la dimensione orizzontale dello SPRITE.

220 Stesso discorso fatto alla riga 210, solo che qui è coinvolta la dimensione verticale.

230 Il risultato dello SPRITE in DATA è gestito dalla riga 380.

240 Per il salvataggio o il caricamento da nastro e per un nuovo schermo o per finire si va alla riga 460.

250-280 A seconda del tasto funzione scelto si mette in R un certo valore, dopodiché si va alla linea 400. Da notare che R è il numero di registro del VIC II (in questo caso) e che 33 è per il colore 0 di sfondo, 37 è il registro multicolore 0, 41 è il colore dello SPRITE N. 2 e 38 è il registro multicolore 1.

290 Stampa il cursore. In questo caso R è la locazione attuale del cursore, C è il carattere che c'è sotto il cursore messo in REVERSE (sommandogli 128).

300-320 Calcolo dello SPRITE e cioè dei valori da usare nel POKE per costruire lo SPRITE. Sappiamo che tutti i valori della griglia sono nella matrice A, che contiene un 49 se è stato premuto il colore 1, 50 per il colore 2 e 52 per il colore 3. Si tratta ora di prendere 8 elementi orizzontali della griglia e trasformarli in un byte, considerando anche il fatto che gli elementi devono essere presi in coppia. Viene letto un elemento, gli viene sottratto 48 in modo tale che il valore sia nuovamente 0, 1, 2, o 3 e viene messo in Q. Poi (linea 320) viene costruito il valore da mettere nella matrice B, utilizzando l'elevazione a potenza perché, come ben sapete, i valori dei bit in un byte sono tutti in potenza di 2.

Questo viene fatto per tutti gli elementi della matrice A, che sono usati per riempire la matrice B con i 63 valori necessari. Subito dopo (sempre linea 320) vengono scritti in memoria (a 831+X) i dati dello SPRITE appena costruito, e si ritorna alla lettura della tastiera.

330-360 Subroutine (chiamata alla linea 460) che ristampa il comando battuto, chiede conferma, mette la risposta in N\$ e posiziona il cursore.

370 Subroutine (chiamata alla riga 390) che si rifà alla precedente, ma chiedendo stavolta quando si vuole continuare.

380 Stampa della matrice B, contenente tutti i valori necessari per costruire lo SPRITE. La forma «strana» dei comandi di stampa è dovuta al fatto che si vogliono stampare 7 righe di 9 dati ciascuna.

390 Dopo aver stampato la matrice, va a 370 per stampare il messaggio «CONTINUA» e dopo va a 60 per ristampare lo schermo con la matrice.

400-450 Qui troviamo la gestione della colorazione dello SPRITE (gestita con i tasti funzionali). Quando si arriva qui R contiene il numero del registro del VIC II interessato (vedere commento alle righe 250-280). Per prima cosa si legge il contenuto attuale del registro e lo si mette in C, dopo avergli dato un AND con 15 perché il valore netto non può e non deve superare 15. Poi lo si incrementa di 1 (cambiando quindi il codice del colore) ponendo attenzione a che non superi il valore massimo di 15 (in questo caso viene messo a zero). Alla linea 410 viene scritto, nel registro selezionato, il nuovo valore e si comincia a posizionare il cursore sulla prima riga dei comandi (quella col comando di cancellazione). Le linee da 410 a 440 servono giusto a posizionarsi sullo schermo di fianco alla scritta «CANC» se si è cambiato il colore dello schermo oppure di fianco alle scritte 1, 2 e 3 se si è cambiato un colore dello SPRITE. Il nuovo colore viene «scritto» alla linea 450.

Remarks Listato - Sprite Editor

460-480 Qui comincia la gestione dei rimanenti 4 comandi. Se il comando è un «N» (Nuovo schermo) si va alla linea 20 per rifare tutto. Se il comando riguarda il salvataggio o la lettura o la fine si chiede conferma (perché in ogni caso si distruggerà quello che si è fatto finora) facendo un GOSUB a 330. Se non c'è conferma si riprende il giro. Se c'è conferma, si analizza il comando. Se il comando è «F» (fine) si «spegne» lo Sprite e si finisce il programma (linea 480).

490 Se si giunge qui, il comando riguarda il salvataggio o la lettura di uno SPRITE da/su cassetta, per cui si «spegne» lo SPRITE e si chiede il nome, che servirà sia a scriverlo che a leggerlo.

500 Se il comando è «L» (lettura) si fa la OPEN per leggerlo e si va a 520.

510 Se il comando è «S» (scrittura) si scrive tutta la matrice B su nastro e si ricomincia da capo mostrando di nuovo lo SPRITE.

520-550 Continuazione del comando «L» (lettura da nastro). Si comincia col leggere tutti i dati necessari e li si mette nella matrice B. A questo punto bisogna riempire la matrice A, che sappiamo contenere non già i dati di B, ma una loro estrapolazione, in quanto ogni dato (o byte) in B è rappresentato da ben otto dati (o bytes) in A. Ed ecco quindi i calcoli fatti alle linee 530-540, che non fanno altro che convertire un dato in otto dati. Infine si resetta il posizionamento del cursore e si va a mostrare il lavoro fatto.

560-570. Nomi di tutti i colori disponibili, usati per scriverli nella parte «comandi» dello schermo.

appunto a poter indirizzare le posizioni orizzontali da 257 a 320. Questa complessità è necessaria per mantenere una così potente capacità grafica.

Gli Sprite standard (e cioè colorati interamente con un solo colore), possono essere mostrati in ognuno dei sedici colori disponibili. Gli Sprites multicolore invece possono essere colorati con quattro diversi colori per ogni Sprite, ed i colori sono determinati considerando i bit a coppie. Se la coppia è formata da due zeri (00) viene selezionato il colore dello sfondo (cioè non si vede niente), «01» seleziona il colore che è stato posto nel registro multicolore numero 0, «10» il colore che è stato posto nel registro di colore standard per quello Sprite, e «11» il colore che è stato messo nel registro multicolore numero 1. Usando gli Sprites nel modo multicolore, la definizione orizzontale viene un po' a mancare, in quanto avremo dei punti (rispetto agli Sprite monocolori) che sono larghi due volte un punto dello schermo.

Ogni Sprite può essere raddoppiato in larghezza e/o in altezza.

Per gestire «dolcemente» l'entrata e l'uscita degli Sprites sullo schermo, le posizioni possibili per X e Y sono oltre la porzione visibile sullo schermo (512x256 invece che 320x200). In questo modo è possibile vedere «spuntare» uno Sprite lentamente da qualunque parte dello

Listato - Sprite Editor

```

9 REM INIZIALIZZAZIONE GENERALE
10 POKE 53281,6: DIM A(21,24), B(63), AS(15): X=0: Y=0: P=0:
  C=0: S=1039: SI=55311
19 REM INIZIALIZZAZIONE DEL VIC II
20 V=53248: POKE V+21,0: POKE V+23,0: POKE V+29,0: RESTORE
  :FOR Y=0 TO 15: READ AS(X): NEXT
29 REM RIFILPIMENTO MATRICI A E B
30 PRINT"(CLR)": FOR R=1 TO 21: FOR C=1 TO 24: A(R,C)=46:
  NEXT: NEXT: FOR Y=1 TO 63: B(Y)=0: NEXT
39 REM POSIZIONAMENTO SPRITE N.2
40 POKE V+4,60: POKE V+5,200: POKE 2042,13: POKE 7+37,0:
  POKE V+41,14: POKE V+38,1
50 FOR X=1 TO 63: POKE 831+X,B(X): NEXT: POKE V+21,2: POKE
  V+28,4
59 REM STAMPA DELLA LISTA DEI COMANDI
60 PRINT"(CLR) (CG) EDITOR DI SPRITE (CG)"
70 PRINT"← CANC."
75 PRINT"1 MC 0 - NERO"
80 PRINT"2 SC - BLU CHI."
85 PRINT"3 MC 1 - BIANCO"
90 PRINT"= CALCOLA SPRITE"
95 PRINT"X CAMBIA IN 'X'"
100 PRINT"Y CAMBIA IN 'Y'"
105 PRINT"R DATI DEL BASIC"
110 PRINT"N NUOVO SCHERMO"
115 PRINT"S PA' IL SAVE"
120 PRINT"L PA' IL LOAD"
125 PRINT"P FINE"
129 REM DISEGNO DELLA GIOGLIA
130 Y=0: FOR R=1 TO 21: FOR C=1 TO 24: Y=Y+1: POKE S+Y,A(R,C)
  : POKE SI+Y,14: NEXT: Y=Y+16: NEXT
140 X=1: Y=1: GOTO 290
149 REM LETTURA COMANDO BATTUTO
150 GET AS: IF AS="" THEN 150
160 P=S+X+(Y-1)*40: C=A(Y,X): POKE P,C: POKE R+1,C
170 IF AS="(C)" THEN Y=Y+1: IF Y>21 THEN Y=1
175 IF AS="(CA)" THEN Y=Y-1: IF Y<1 THEN Y=21
180 IF AS="(CD)" THEN X=X+2: IF X>24 THEN X=1
185 IF AS="(CS)" THEN X=X-2: IF X<1 THEN X=23
190 IF AS="←" THEN A(Y,X)=46: A(Y,X+1)=46
195 IF AS">" AND AS<"4" THEN P=49+VAL(AS): A(Y,X)=":
  A(Y,X+1)=4
200 IF AS="" THEN 300
210 IF AS="X" THEN POKE V+29,ABS(PEEK(V+29)-4)
220 IF AS="Y" THEN POKE V+23,ABS(PEEK(V+23)-4)
230 IF AS="R" THEN 380
240 IF AS="N" OR AS="S" OR AS="X" OR AS="F" THEN 460
250 IF AS="(P1)" THEN R=33: GOSUB 400
260 IF AS="(P3)" THEN R=37: GOSUB 400
270 IF AS="(P5)" THEN R=41: GOSUB 400
280 IF AS="(P7)" THEN R=39: GOSUB 400
289 REM POSIZIONAMENTO CURSORE
290 P=S+X+(Y-1)*40: C=(A(Y,X)+128): POKE P,C: POKE R+1,C:
  GOTO 150

```


Listato - Sprite Editor

```

299 REM CALCOLO DELLO SPRITE
300 Y=0: FOR R=1 TO 21: FOR X=0 TO 2: Y=Y+1: B(Y)=0: FOR C=1
    TO 7 STEP 2: Q=A(R,X*B+C)-48
310 IF Q<0 OR Q>3 THEN Q=0
320 B(Y)=B(Y)+2*(7-C)*Q: NEXT: NEXT: NEXT: FOR X=1 TO 63:
    POKE 831+X,B(X): NEXT: GOTO 140
329 REM STAMPA RICHIESTA DI CONFERMA DEL COMANDO
330 PRINT"(REV) "AS": SI O NO"
340 FOR X=1 TO 10: GET N$: NEXT
350 GET N$: IF N$="" THEN 350
360 PRINT"(CA)      (16 spazi) (CA)": RETURN
370 PRINT"(REV) CONTINUA": GOTO 340
379 REM STAMPA MATRICE B (DATI DELLO SPRITE)
380 PRINT"(CLR)": FOR X=1 TO 7: PRINT"DATA": FOR Y=1 TO 9:
    PRINT B((X-1)*9+Y),"": NEXT
390 PRINT"(CS) ":NEXT: PRINT: GOSUB 370: GOTO 60
399 REM GESTIONE DELLA COLORAZIONE DELLO SPRITE
400 C=PEEK(V+R) AND 15: C=C+1: IF C>15 THEN C=0
410 POKE V+R,C: PRINT"(HOME). (3 CG)": IF R=33 THEN 450
420 PRINT"(CG)": IF P=37 THEN 450
430 PRINT"(CG)": IF P=41 THEN 450
440 PRINT"(CG)":
450 PRINT"(7 CD) "AS(C) " ":RETURN
459 REM GESTIONE COMANDI RIMANENTI
460 GOSUB 330: IF N$<>"S" THEN 290
470 GET N$: GET N$: IF AS="N" THEN 20
480 IF AS="F" THEN POKE V+21,0: POKE V+28,0: PRINT"(4 CG)":
    END
489 REM GESTIONE COMANDI 'L' O 'S'
490 PRINT"(CLR)": POKE V+21,0: INPUT"NOVE DELLO SPRITE": N$:
    PRINT
500 IF AS="L" THEN OPEN 1,1,0,N$: GOTO 520
510 OPEN 1,1,1,N$: FOR X=1 TO 63: PRINT#1,B(X): NEXT: CLOSE 1:
    GOTO 50
519 REM CONTINUAZIONE COMANDO 'L'
520 FOR X=1 TO 63: INPUT#1,B(X): NEXT: CLOSE 1: PRINT"(CG)
    CALCOLO LA MATRICE"
530 Y=0: FOR R=1 TO 21: FOR X=0 TO 2: Y=Y+1: FOR C=2 TO 8
    STEP 2: Q=X*B+C: P=2*(8-C)
540 S=P(Y) AND (P*3): A(P,Q)=46: A(P,Q-1)=S/P+48
550 NEXT: NEXT: NEXT: S=1029: GOTO 50
559 REM NOXI DEI COLORI, DA CARICARE IN AS (15)
560 DATA NERO., BIANCO, ROSSO, CYAN, PORPORA, VERDE., BLU.,
    GIALLO
570 DATA ARANCIO, MARRONE, ROSSO C, GRIGIO1, GRIGIO2,
    VERDE C, BLU CHI., GRIGIO3

```

Lista variabili - Sprite Editor

Lista delle variabili

A (21,24) - Matrice contenente la raffigurazione della griglia mostrata sullo schermo
B (63) - Matrice contenente i dati per costruire lo SPRITE
AS (15) - Matrice contenente i nomi dei colori disponibili
X,Y - Posizioni orizzontale e verticale del cursore. Usati anche con altri significati.
R - Riga della matrice A usata anche con altri significati (registro VIC II, etc.)
C - Colonna della matrice A usata anche con altri significati.
S - Indirizzo della memoria video per gestire il disegno della griglia
SI - Indirizzo della memoria colore, usata per gestire il disegno della griglia.
Q - Valore del colore selezionato (da 0 a 3)
V - Indirizzo del VIC II
AS - Comando battuto

schermo.

Prima ho menzionato le priorità. Gli Sprites, di per sé, hanno delle priorità fisse uno rispetto l'altro: lo Sprite 0 ha una priorità più alta dello Sprite N. 1, l'1 ha una priorità più alta dello Sprite N. 2 e così via. Però ogni Sprite ha la priorità nei confronti dei dati sullo sfondo (testo o sfondo vuoto) che può essere scelta a piacere. Da notare che gli oggetti in priorità più alta «passano sopra» a quelli in priorità più bassa, senza «cancellarli».

Gli scontri sono avvertiti dal VIC II, e i bits appropriati sono messi a 1 in due registri. I bit messi a uno in questa maniera, saranno ripristinati al valore iniziale di zero appena il vostro programma li avrà letti. In questo caso l'intero registro viene azzerato.

C'è un registro per la collisione fra due Sprites, ed uno per la collisione tra Sprite e sfondo.

Alcune limitazioni possono essere evitate usando tecniche di programmazione più sofisticate. Per esempio è possibile avere più di otto Sprites sul video mostrati contemporaneamente, usando una tecnica detta «raster interrupt». Ancora, potete avere (visto l'abbondanza di memoria) molti Sprites definiti in memoria, e potete cambiare i puntatori a seconda di quale vi serve.

Creazione di uno Sprite

Per creare uno Sprite, dovete disegnarlo in una griglia di 24x21 punti. Dopodiché dovete convertire l'insieme dei punti in una riga in tre bytes di dati, usando il codice binario. Per ogni bytes si sommano i valori dei singoli bits messi a uno. Tutto questo è spiegato, abbastanza chiaramente, nel manuale del 64. Da tutto ciò si rileva che creare gli Sprites non è eccessivamente difficile, una volta presa la mano, ma terribilmente tedioso. Il programma oggetto di questo articolo è appunto un «Edi-

tor» di Sprites e cioè un «attrezzo» per costruire velocemente gli Sprites. I comandi che accetta sono tutti attuati con un tasto: potete visualizzare lo Sprite che state costruendo, fare delle modifiche se volete, farvi mostrare i dati da usare poi nel vostro programma, scriverlo su nastro od eventualmente rileggerlo per fargli ulteriori modifiche. E scusate se è poco. Una volta fatto partire il programma, tutti i comandi disponibili sono stampati sulla parte sinistra dello schermo. Sulla parte destra invece ci sarà una griglia di 24x21 punti, che è usata per creare/modificare lo Sprite. Per muovere il cursore (che sarà posizionato inizialmente in alto a sinistra) usate i soliti tasti di controllo del cursore. Se volete «accendere» un punto nello Sprite, premete i tasti 1, 2 o 3. A questi tasti sono associati tre colori (inizialmente nero, azzurro e bianco) che potete cambiare premendo i tasti funzionali 3, 5, e 7 rispettivamente.

Mano a mano che cambiate i colori vedrete che anche le descrizioni dei tasti 1, 2 e 3 cambieranno. MC di fianco ai tasti 1 e 3 significa Multi-Colore, mentre SC di fianco al tasto 2 significa Singolo Colore: si riferiscono ovviamente ai registri del VIC II. È importante ricordarsi di ciò perché se vorrete gli stessi colori nel programma che userà lo Sprite, dovrete mettere gli stessi colori nei registri corrispondenti (Registri multicolore 0 e 1 e registro di colore dello Sprite).

Se invece volete cambiare il colore di tutto lo schermo potete farlo battendo il tasto funzione F1. Se volete cancellare una coppia di punti precedentemente accesa, lo potete fare con il tasto «». In qualunque momento potete vedere lo Sprite come sarà mostrato nelle sue dimensioni reali premendo il tasto «=»: lo Sprite vi comparirà nella parte in basso a sinistra dello schermo. Se cambiate i colori vedrete che anche il

«piccolo» Sprite cambierà colore. Ricordatevi comunque di battere il tasto di calcolo dello Sprite (l'uguale) prima di visualizzare i dati per la sua futura costruzione (con il tasto «B») o prima di salvarlo su nastro (con il tasto «S»). Questo perché solo quando voi battete «=» il programma si calcola tutte le informazioni per creare lo Sprite, e non mentre voi lo disegnate nella griglia.

Anche se fate delle modifiche al disegno, queste non saranno riportate sul «piccolo» fino a che non batterete l'«=». Le altre cose che potete fare sono l'espansione orizzontale del «piccolo» Sprite, come pure quella verticale (battendo rispettivamente X e/o Y). Per ricominciare a disegnare un altro Sprite battete il tasto «N» (nuovo schermo) e azzererete tutti i dati. Battendo invece «L» caricherete uno Sprite già fatto da nastro mentre il tasto «F» serve a far finire il programma.

Note sul listato

Nel listato del programma CG significa «Cursore Giù»; CA invece vuol dire Cursore in Alto, CD cursore a destra e CS cursore a sinistra F1, F3, F5, F7 sono i rispettivi tasti funzionali.

«CLR» vuol dire Clear Screen, REV vuol dire REVERSE ON. Qualche volta davanti ai tasti di movimento cursore c'è un numero: vuol dire di battere quel tasto un certo numero di volte. Per esempio: (3 CG) vuol dire di battere tre volte il tasto di Cursore. Un'ultima cosa: le ultime due righe del programma contengono i nomi dei colori, e dopo alcuni nomi c'è un punto: battetelo (non è un errore di stampa). Serve a fare in modo che quando cambiate colore non resti, sotto il nuovo, qualche rimasuglio del nome del colore che c'era prima. Per la spiegazione dettagliata del programma vi rimando alle REMarks. Auguri.

Roll up e roll down per Vic 20

Questo piccolo programma vi permetterà di listare i vostri programmi ed eventualmente di ritornare su una linea appena listata.

L'editing da schermo essendo di soli 22 caratteri richiede, se il programma è abbastanza lungo, parecchio tempo e molta noia mantenendo la possibilità di fare altri errori. Un buon debug si fa o con listato su carta, o avendo a disposizione la possibilità di listare in modo bidirezionale: bene con questo programmino anche voi che possedete il Vic 20 potrete farlo.

La linea 1001 determina l'indirizzo della locazione di memoria di partenza (GO) per ogni particella di memoria del VIC. La linea 1002 calcola il numero delle linee (LN) del programma da listare. La linea 1003 prepara il video per il listato della linea, dopodiché continua il programma. Appena terminata la lista, il programma termina, e questo è gestito dalla linea 1004 che controlla la lista e continua in sequenza dato il comando go-

to 1010. In linea 1010 e 1030 sono le linee che vi danno la possibilità di rollappare o rollaunare (belli no!), premendo o il segno del minore (<) o il segno del maggiore (>) per andare rispettivamente in su o in giù.

Attenzione alla linea 1100 che controlla il prossimo O in Basic, ossia la prossima fine linea basic e che quindi vi rimanda a calcolare il prossimo numero di linea. La linea 1200 controlla in modo routine la fine della linea precedente per potervi eventualmente ritornare. Per poter usare bene questo mini programma dovrete quindi eliminare gli O negli indirizzi che determinano il numero di linea, non amettendovi uno O in questa locazione di memoria.

Onde evitare di ribattere sempre questo seppur corto programma al momento di una LOAD ad un programma applicativo, eseguite:

CLR: PRINT PEEK (45): PRINT PEEK (46)

Battete quindi la linea immediatamente seguente

POKE 43, PEEK (45)-2: POKE 44, PEEK (46)

Tale linea vi permette di spostare l'inizio del Basic a due byte prima della fine del

Listato - Roll up

```
1000 PEM *** SUBTULST ***
1001 GO=PEEK(44)+256-PEEK(43)-1
1002 LN=PEEK(GO+3)+PEEK(GO+4)+256
1003 PRINT " GOTO 1010 :PRINT
      "LIST" :LH1
1004 POKE53,19:POKE32,17
1005 POKE63,31:POKE63,13
1006 POKE65,19:POKE63,13
1007 POKE198,61:END
1010 IF PEEK(197)=60 THEN 1100
1020 IF PEEK(197)=62 THEN 1200
1030 GOTO 1010
1100 IF PEEK(GO+5)=3 THEN GO=GO+1
1105 GOTO 1100
1110 GO=GO+5 :GOTO 1002
1200 GO=GO-1:IF PEEK(GO)=0 AND
      PEEK(GO+4)<>0 AND
      PEEK(GO+3)<>0 THEN 1002
1210 GOTO 1200
```


programma (sarà necessario un nullo o uno 0 per iniziare a caricare un nuovo programma).

Ora caricate il vostro programma che chiamerete «SUGIULST» (o se no chiamatelo un po' come vi pare), resettate i puntatori basic con : POKE 42,1; POKE 44,16 per il vostro VIC inespanso. Dopodiché cominciate ad editare battendo RUN 1000.

Avrete la possibilità di verificare linea per linea il vostro programma senza doverlo listare su carta dopo ogni modifica.

Il «Simon» sul C64

Vi proponiamo un programma inviato da Roberto Anglesio di Torino: una interessante realizzazio-

ne su C64 del famosissimo gioco «Simon».

Il gioco originale è costituito da un apparecchio circolare capace di emettere delle sequenze di suoni in tonalità differenti. Una volta ascoltata la sequenza il giocatore deve riprodurla esattamente, agendo su uno dei quattro settori circolari, colorati diversamente e posti sulla parte alta dell'apparecchio. All'inizio «Simon» emette una nota sola ma, a mano a mano che il giocatore riesce a riprodurre i vari suoni senza sbagliare, il numero di note emesse si incrementa di una.

Tutto questo è stato riprodotto sul Commodore ed il gioco può dirsi identico all'originale per quanto concerne la sua meccanica.

I quattro settori circolari di Simon sono sostituiti dai quattro tasti funzione del

Convenzione usata per la scrittura dei caratteri grafici - Simon

(cls)	= CLR HOME + SHIFT
(n spazi)	= Numero di spazi
()	= 1 spazio
(176.96 ecc)	= Rappresenta il codice ASCII dei caratteri, per visualizzare il carattere basta impostare: ?CHRS(n.)160 corrisponde allo spazio
(home)	= CLR HOME
(Cs)	= Corsore a sinistra
Solo alle linee 90, 100, 120, 140, 280 e 290 ho disegnato direttamente il carattere (Q+Shift e W+Shift)	

Variabili usate nel programma Simon

A (4,4)	= Tabella corrispondente alla tastiera
G	= Indica a quale giocatore va riferita la mossa
G1\$,G2\$	= Nomi dei giocatori
M\$	= Lettera corrispondente alla mossa
M	= Codice ASCII relativo a M\$
S	= Corrisponde al valore (pedina) da inserire nella tabella A
L	= Riga o colonna corrispondente alla lettera di input
C,D,E,F	= Valori usati per aggiornare lo schermo
H	= Indica il tipo di verifica da applicare alla somma
A1	= Somma parziale
T	= Somma
J\$	= Nome del vincitore
A,B	= Coordinate della scacchiera
P	= ASCII delle lettere da scrivere durante l'inizializzazione dello schermo

Note a «Simon»

Nelle istruzioni di stampa o di definizione di stringhe le dicititure tra parentesi indicano rispettivamente:

- CLR/HOME = tasti SHIFT+CLR/HOME
- CRSR G = cursore giù
- CRSR D = cursore a destra
- CTRL n. = tasti CONTROL+n. dove n. è il tasto numerico specificato
- C=x = tasti SIMBOLO COMMODORE+x dove x è il tasto specificato
- HOME = tasto CLR/HOME da solo
- F1, F3, F5, F7 = tasti di funzione

Se le combinazioni di tasti sopra descritte devono essere digitate più di una volta, è indicato subito dopo l'apertura della parentesi il numero di volte relativo.

Nella riga 130 i caratteri (C) devono essere digitati così come sono.

Descrizione del programma «Simon»

Righe 10-20 viene preparata la matrice contenente le diverse combinazioni di note; tali note vengono determinate casualmente

Righe 30-120 costituzione stringhe per il titolo animato

Righe 130-160 intestazione e animazione titolo

Riga 165 viene azzerato il buffer di tastiera

Riga 170 aspetta che un tasto venga premuto per incominciare

Righe 175-176 informa sul modo di giocare

Righe 180-190 pulisce lo schermo e setta il SID

Righe 200-220 esegue la sequenza di suoni

Riga 225 azzerata il buffer di tastiera

Righe 230-240 richiede al giocatore la sequenza di suoni

Righe 250-280 assegna un valore a seconda del tasto premuto

Riga 290 se il tasto premuto non è uno dei quattro ammessi torna a 240

Riga 300 se il tasto premuto corrisponde ad un suono sbagliato, va alla routine di errore

Righe 310-320 esegue la sequenza impostata dal giocatore

Righe 1000-1300 a seconda del suono da eseguire assegna diversi valori per il colore dello schermo, la frequenza della nota, il numero corrispondente al tasto di funzione da premere e il colore di tale numero

Riga 1400 emette un suono che segnala l'errore

Riga 1410 stampa il messaggio relativo al numero di note che si sono riuscite a mettere in sequenza

Riga 1420 pulisce lo schermo, punta ad una nuova sequenza di note e torna all'inizio del gioco

Righe 1500-1520 routine che emette il suono, colora lo schermo, stampa il numero relativo al tasto di funzione da premere.

C64; inoltre, poiché detti settori (nel gioco originale) si illuminano per facilitare il compito del giocatore, lo schermo del computer si colora diversamente a seconda della nota emessa e, al centro dello schermo stesso, appare un numero che corrisponde al tasto funzione in-

teressato.

Quando il giocatore sbaglia, il computer emette un suono che lo avverte dell'errore e gli indica poi il numero di note che è riuscito a mettere in sequenza.

Attenzione però: il giocatore non può pensare troppo a lungo perché dopo un certo

Listato Simon

```

5 REM + SIMON PER COMMODORE 64 +
6 REM + ROBERTO ANGLESIO 011/616128 +
10 PRINT"(CLR/HOME)(2 CRSR G)(9 CRSR D)PLEASE WAIT....":DIMSE(50,30)
20 FORK=1TO50:FORJ=1TO30:SE(K,J)=INT(RND(0)*4)+1:NEXTJ:NEXTK
30 F$(1)="(CTRL 8)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)
(C=B)(CTRL 0)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
40 B$(1)="(C=B)(5 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CTRL 9)(2 C=B)(CTRL 0)
(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CRSR D)(C=B)"
50 C$(1)="(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)(C=B)
(CTRL 0)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CRSR D)(2 C=B)"
60 D$(1)="(3 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)
(C=B)(CTRL 0)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
70 E$(1)="(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 9)(C=B)
(CTRL 0)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
80 F$(2)="(CTRL 6)(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CTRL 0)(C=B)
(CTRL 9)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
90 B$(2)="(CTRL 9)(C=B)(5 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CTRL 0)(2 C=B)
(CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(2 C=B)(CRSR D)(C=B)"
100 C$(2)="(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 0)
(C=B)(CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CRSR D)(2 C=B)"
110 D$(2)="(CTRL 9)(3 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(CTRL 0)(C=B)
(CTRL 9)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
120 E$(2)="(CTRL 9)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)(2 CRSR D)(CTRL 0)
(C=B)(CTRL 9)(2 CRSR D)(4 C=B)(2 CRSR D)(C=B)(2 CRSR D)(C=B)"
130 POKE53280,6:PRINT"(CLR/HOME)(17 CRSR G)(10 CRSR D)(C) DER BESTE 1984":
I=2:D=1
140 PRINT"(6 CRSR G)(13 CRSR D)(CTRL 9)(CTRL 4) HIT ANY KEY (C=7)(CTRL 0)";
150 PRINT"(HOME)(4 CRSR G)":D=D-I:I=I+D
160 PRINTSPC(6)F$(I):PRINTSPC(6)B$(I):PRINTSPC(6)C$(I):PRINTSPC(6)D$(I):
PRINTSPC(6)E$(I)
165 POKE 198,0
170 FORI=1TO30:GETA$:IFA$=""THENNEXT:GOTO150
175PRINT"(CLR/HOME)(2 CRSR G)(5 CRSR D)PER SUONARE USARE I TASTI DI"
176 PRINT"(CRSR G)(5 CRSR D)FUNZIONE 'F1' 'F3' 'F5' 'F7':FORT=1TO2000:NEXTT
180 PRINT"(CLR/HOME)":SI=54272:FH=SI+1:W=SI+4:A=SI+5:H=SI+6:L=SI+24:X=1:J=1
190 POKEL,15:POKEA,0:POKEH,160
200FORP=1TOJ:RO=1
210 ONSE(K,P)GOSUB1000,1100,1200,1300
220 NEXTP
225 POKE198,0
230 FORP=1TOJ
240 FORR=1TO500:GETA$:IFA$=""THENNEXT:GOTO1400
250 IFA$="(F1)"THENN=1:GOTO300
260 IFA$="(F3)"THENN=2:GOTO300
270 IFA$="(F5)"THENN=3:GOTO300
280 IFA$="(F7)"THENN=4:GOTO300
290 NEXTR
300 IFSE(K,P)<>NTHEN1400
310ONNGOSUB1000,1100,1200,1300
320 NEXTP:FORT=1TO200:NEXTT:J=J+1:GOTO200
1000 SC=7:SU=30:NR=49:SF=0:GOTO1500
1100 SC=5:SU=40:NR=51:SF=0:GOTO1500
1200 SC=2:SU=50:NR=53:SF=1:GOTO1500

```


Listato Simon

```

1300 SC=6:SU=60:NR=55:SF=1:GOTO1500
1400 POKEFH,10:POKESI,50:POKEW,33:FORT=1TO200:NEXTT:POKEW,0
1410 PRINT"(CLR/HOME)(8 CRSR G)          HAI MESSO IN SEQUENZA"J-1"NOTE!":
      FORT=1TO1500:NEXTT
1420 PRINT"(CLR/HOME)":K=K+1:J=1:GOTO200
1500 POKE55795,SF:POKE1523,NR
1505 POKEFH,SU:POKESI,100:POKEW,17:POKE53281,SC:POKE53280,SC:FORT=1TO200:
      NEXTT:POKEW,0
1510 IFRO=1THENFORT=1TO200:NEXTT:RO=0
1520 POKE53281,0:POKE53280,0:POKE1523,32:RETURN

```

intervallo di tempo, se nessuno dei tasti funzione è stato premuto, il computer si comporta come se si fosse commesso un errore e si deve ricominciare.

«Black Box» sul VIC 20

Vi proponiamo una bella soluzione di «Black Box», gioco di riflessione alla scoperta delle leggi della natura, proposto su Personal Computer n. 2, nella versione per Vic 20 dell'amico Fulvio Forti di Novate Milanese. Più che di una soluzione si tratta in realtà di uno studio con tanto di bibliografia (in lingua inglese: E.W. Solomon: Black Box Game Book, Strategy Games Publ., London, 1977), in cui sono riportate tra l'altro le regole originali che noi avevamo lievemente semplificato nel caso di deflessioni sul bordo della scatola nera. Bene, note le leggi di riflessione, deflessione e assorbimento dei raggi che – lanciati all'interno della scatola nera – servono a darvi un'idea della disposizione degli atomi in essa, si tratta soltanto di fare un numero sufficiente di prove per confermare le proprie congetture. Nelle due figure allegate sono riportati esempi di queste leggi nonché la numerazione delle caselle di ingresso dei raggi sulla scacchiera 8x8 su cui è stato programmato il gioco. Per comunicare al Personal

Listato Black Box

[illegible]

la direzione dei propri tentativi è sufficiente infatti indicare un numero compreso tra 1 e 32, essendo da ciò implicito anche il verso prescelto.

Per inserire un asterisco ad indicazione di un atomo della molecola (soltanto al fine di aiutare le proprie considerazioni) bisognerà digitare I: con lo stesso comando si potrà togliere un atomo mal piazzato. Con C infine si chiede al Personal di concludere il gioco, controllando se tutti gli atomi piazzati sono al posto giusto (condizione di vittoria) oppure no.

È appena il caso di dire che il gioco si svolge in solitario contro il Personal che nasconde gli atomi (righe 400-470 del listato), che il programma gira sul VIC 20 in configurazione base e che utilizza un algoritmo di numerazione delle caselle lievemente differente da quello descritto, che però è trasparente al giocatore, in quanto è «tradotto» prima di essere presentato a video. I simboli grafici riportati nel listato sono specificati accanto ad esso.

Strategia per scacchiera con «Othello» sul VIC 20

Di origine - si dice - giapponese, Othello è uno dei più interessanti giochi di strategia per scacchiera di recente invenzione. Noto anche come «Reversi», come tutti i grandi giochi ha il fascino di regole elementari, al punto che la sua pubblicità (in Italia è distribuito dalla Clem Toys) parla di «un minuto per impararlo, una vita per diventare campioni».

Su una normale scacchiera 8x8 si inizia a giocare disponendo al centro quattro pedine in quadrato, due del colore di un giocatore e due del

Listato - Black Box

```

35Ø IFA$="N"THENPRINT"▽":END
36Ø GOTO33Ø
40Ø FORJ=1TO4
41Ø X(J)=INT(RND(1)*8+1)*1Ø+INT(RND(1)*8+1)
42Ø FORK=1TO4
43Ø IFX(J)=S(K)THENK=4:GOTO41Ø
44Ø NEXTK
45Ø S(J)=X(J)
46Ø NEXTJ
47Ø RETURN
50Ø PRINT"▽"TAB(2)"** SCATOLA NERA **"
51Ø PRINT"Q 33322222"
52Ø PRINT" 21Ø98765"
53Ø PRINT"1.....24"
54Ø PRINT"2.....23"
55Ø PRINT"3.....22"
56Ø PRINT"4.....21"
57Ø PRINT"5.....2Ø"
58Ø PRINT"6.....19"
59Ø PRINT"7.....18"
60Ø PRINT"8.....17"
61Ø PRINT" 91111111"
62Ø PRINT" Ø123456"
63Ø PRINT"SQQTAB(13)"ENT USC"
64Ø PRINTTAB(11)" | | |",
65Ø FORK=1TO15
66Ø PRINTTAB(11)" | | |",
67Ø NEXTK
68Ø PRINTTAB(11)" | | |",
69Ø RETURN
70Ø PRINT"SQQQQQQQQQQQQQQQQQQQQ";
71Ø FORK=ØTO43
72Ø POKE8142+K,32
73Ø NEXTK
74Ø RETURN
80Ø GOSUB7ØØ:INPUT"COORDINATA ORIZ. ";H$
81Ø H=VAL(H$)
82Ø IFH>8ANDH<17THENH=H-8:GOTO85Ø
83Ø IFH>24ANDH<33THENH=33-H:GOTO85Ø
84Ø GOTO8ØØ
85Ø GOSUB7ØØ:INPUT"COORDINATA VERT. ";V$
86Ø V=VAL(V$)
87Ø IFV>ØANDV<9THEN9ØØ
88Ø IFV>16ANDV<25THENV=25-V:GOTO9ØØ
89Ø GOTO85Ø
90Ø GOSUB7ØØ:INPUT" SIMBOLO (*,.)";R$
91Ø R=ASC(R$)
92Ø IFR=42ORR=46THEN94Ø

```


Listato - Block Box

```

93Ø GOTO9ØØ
94Ø POKEHV+H+V*22,R
95Ø POKEC+H+V*22,6
96Ø RETURN
1ØØØ IFE<9 THENP=E*1Ø:A=1:B=1Ø:GOTO1Ø4Ø
1Ø1Ø IFE>8 ANDE<17 THENP=E+82:A=-1Ø:B=1:GOTO1Ø4Ø
1Ø2Ø IFE>16 ANDE<25 THENP=(26-E)*1Ø-1:A=-1:B=1Ø:GOTO1Ø4Ø
1Ø3Ø P=33-E:A=1Ø:B=1
1Ø4Ø X=A:Y=B
1Ø5Ø FORJ=1TO4
1Ø6Ø IFP+X=S(J) THENU=Ø:RETURN
1Ø7Ø NEXTJ
1Ø8Ø FORJ=1TO4
1Ø9Ø IFP+X+Y<>S(J) THEN112Ø
11ØØ J=4:GOSUB124Ø:IFF=1 THENRETURN
111Ø GOTO119Ø
112Ø NEXTJ
113Ø FORJ=1TO4
114Ø IFP+X-Y<>S(J) THEN117Ø
115Ø J=4:GOSUB124Ø:IFF=1 THENRETURN
116Ø A=Y:B=ABS(X):GOTO1Ø4Ø
117Ø NEXTJ
118Ø GOTO122Ø
119Ø FORJ=1TO4
12ØØ IFP+X-Y=S(J) THENA=-X:GOTO1Ø4Ø
121Ø A=-Y:B=ABS(X):GOTO1Ø4Ø
122Ø P=P+X:GOSUB124Ø:IFF=1 THENRETURN
123Ø GOTO1Ø4Ø
124Ø IFP/1Ø-INT(P/1Ø)=Ø THENF=1:U=P/1Ø:RETURN
125Ø IFP>9Ø THENF=1:U=P-82:RETURN
126Ø IF(P+1)/1Ø-INT((P+1)/1Ø)=Ø THENF=1:U=26-5P+1)/1Ø:RETURN
127Ø IFP<9 THENF=1:U=33-P:RETURN
128Ø F=Ø:RETURN

```

Simboli grafici utilizzati nel listato Black Box

<u>Ø</u>	SHIFT + CLR/HOME
<u>Q</u>	CRSR in basso
<u>J</u>	CRSR a destra
<u>S</u>	HOMÉ

suo avversario. La caratteristica di queste pedine è di essere bifronti, cioè colorate su due lati con i colori dei due contendenti (come certi biscotti che si mangiavano da bambini...). La «mossa» consiste nel deporre una pedina del proprio colore su

una casella libera, a patto che «prenda in mezzo» almeno una pedina avversaria, che viene immediatamente girata e cambia dunque colore. Se non è possibile una mossa di questo tipo si è costretti a «passare». Per controllare se una mossa fa

cambiare colore a qualche pedina avversaria, bisogna esplorare tutte le direzioni (anche diagonali!) a partire dalla posizione giocata, e vedere se c'è qualche pedina (o gruppo di pedine) avversaria che risulti presa in mezzo. È chiaro perciò che i quattro

angoli della scacchiera sono posizioni di tutta sicurezza (e di grande interesse strategico), perché è impossibile far cambiare ad essi colore. Il gioco prosegue in maniera imprevedibile, perché ad ogni mossa cambiano colore molte pedine (TUTTE quelle che soddisfano le regole) e tra un colpo di scena e l'altro si arriva a riempire tutta la scacchiera, oppure ad una posizione in cui nessuno dei due avversari dispone di mosse valide. Il risultato finale, allora, è dato dal numero di pedine del colore dei due contendenti: si va da uno schiacciante 64-0 (63-1, e così via) ad un risicato 33-31 o addirittura alla parità 32-32.

Non pretendiamo certo qui di esaurire la teoria di Othello, che tuttora vede migliaia di giocatori impegnati in tutta Italia ad allenarsi per il campionato italiano che apre le porte ai mondiali (dominati dai giapponesi, ma con brillanti piazzamenti dei nostri rappresentanti): vi suggeriamo soltanto, se dopo un apprendistato su questo gioco vi interessa l'attività agonistica, di mettervi in contatto con la Clem Toys a Recanati (AN).

Sul listato dell'infaticabile Vincenzo Leto di Pisa c'è poco da dire: si gioca in due avversari umani (interessante sarebbe pure giocare contro il computer, ma è un affar serio da programmare), scegliendo il colore delle proprie pedine. Le mosse vanno comunicate alla macchina specificando le coordinate della casella «di arrivo» della pedina. Il vantaggio di giocare a video anziché su una scacchiera sta tutto nel fatto che i controlli sulla validità della mossa, ma soprattutto le pedine avversarie da «girare» sono appannaggio del Personal, laddove molti principianti commettono errori che falsano la partita. Alla fine, la vittoria di uno dei due giocatori è celebrata visivamente e acusticamente per mezzo della «Bourrée» di J.S. Bach. (L.F.)

Listato Othello

```

1 POKE36879,110:PRINT"*****HELLO**"
10 PRINT"*****I-V.LETO"
15 POKE36878,15:FORI=1TO27:READA$:POKE36875,A$:FORJ=1TO250:NEXTJ,I:GOTO1
20 DATA237,239,225,225,239,237,217,217,237,239,207,212,217,237,235,232,207,228
25 DATA225,223,195,201,207,201,195,223,219
30 DIMC$(7),SCX(2,3)
35 FORI=0TO9:FORJ=0TO9:SCX(I,J)=-1:NEXTJ,I:D=38512
40 C$(1)="M":C$(2)="N":C$(3)="L":C$(4)="K":C$(5)="H":C$(6)="G":C$(7)="F":D#=3
45 FORI=15TO9STEP-1:POKE36878,I:NEXT:POKE36879,20
50 INPUT"*****1-NOME":A$:GOSUB520
55 IFLEN(A$)>10THENA$=LEFT$(A$,10)
60 GOSUB560:C=HX:IFC<10RC>7THEN60
65 A#=C$(C)+": "+A$:C$(C)=A$:INPUT"*****2-NOME":B$:GOSUB520
70 IFLEN(B$)>10THENA$=LEFT$(B$,10)
75 GOSUB560:D=HX:IFD<10RD>7ORD=2742476
80 B#=C$(D)+": "+B$:IFC=1THENC=0
85 IFD=1THEND=3
90 Q$="|||||":R$="|||||":PRINT"*****"
95 FORI=1TO7:PRINTR$:PRINTQ$:NEXTI:PRINTR$:PRINT"*****"
100 PRINT"*****1 2 3 4 5 6 7 8 9 10 *****"
105 POKE36875,0:POKE36876,15
110 FORI=4TO5:SCX(I,1)=D:SCX(I,9-1)=C:NEXTI
115 FORI=0TO22STEP2:FORJ=5TO17STEP2:POKE36875,J+22%*I:POKE36876,J+22%*I+16:NEXTJ,I
120 POKE36875,0:POKE36876,17:POKE36875,D:POKE36876,D:POKE36875,0:POKE36876,16:POKE36875,0:POKE36876,1
125 FORH1=1TO60
130 FORJ=77TO7723:POKEJ,02:NEXTJ:PRINT"*****100"
135 GETW$:GOSUB530:IFW$=""THEN135
140 W=VAL(W$):IFW<10RND8THEN135
145 D1=7813+(W-1)*44:POKED1,PEEK(D1)+128
150 GETZ$:GOSUB540:IFZ$=""THEN135
155 Z=VAL(Z$):IFZ<10RND8THEN150
160 D2=77+(Z-1)*2:POKED2,PEEK(D2)+128
165 IFSCX(W,Z)<0-1THENGOTO550
175 FORL=W-1TOW+1:FORK=Z-1TOZ+1:IFL=WRNDRK=ZTHEN185
180 IFSCX(L,K)<0-1THEN190
185 NEXTK,L:GOTO550
190 SCX(W,Z)=C:POKEU+2*Z+44*W-23,C:E=-1:C=G:D=G:GOSUB710:FORI=W-1TOW+1:FORJ=Z-1TOZ+1
195 IFI=WRNDRJ=ZTHENE=1:GOTO210
200 IFSCX(I,J)=DTHENGOSUB600
205 G=G+1
210 NEXTJ,I:P$=A$:A$=B$:B$=P$:G=C:C=D:D=G:GOSUB555:NEXTH1:RESTORE
215 K=0:FORI=1TO9:FORJ=1TO8:IFSCX(I,J)=DTHENK=K+1
220 NEXTJ,I:FORF=7705TO7722:POKEF,32:NEXT:PRINT"*****"A$==">"K"*****B$==">"
225 IFK=32THEN245
230 IFK<32THENC=22:C=1
235 E=1:FORI=1TO27:FORJ=36425TO38442:POKEJ+0%,C:NEXTJ
240 READA$:POKE36875,A$:P=C:C=E:E=P:FORH1=1TO60:NEXTH1:PRINT"*****1 2 3 4 5 6 7 8 9 10 *****"
245 PRINT"*****PARTITA PATTA *****"
250 PRINT"*****CONTINUE (Y/N)? *****"
255 GETL$:IFL$=""THEN255
260 IFL$="C"THEN255
265 IFL$="N"THENPRINT"*****1-2-3-4-5-6-7-8-9-10 *****":POKE36879,110:POKE36878,0:END
270 GOTO255
510 POKE36878,15:POKE36874,120+INT(RND(1)*127):RETURN
520 PRINT"*****":FORI=1TO7:PRINTTAB(4-I)*3+2;C$(1,2):NEXTI:PRINT"*****"
530 POKE36875,2:FORL=1TO22:NEXT:POKE36876,1:FORL=1TO22:NEXT:RETURN
540 POKE36875,2:FORL=1TO200:NEXT:POKE36876,1:FORL=1TO200:NEXT:RETURN
550 PRINT"*****MUCH ERROR, I-10001720:FORL=1TO188:NEXT:GOSUB555:PRINT"*****"

```


Listato Othello

[illegible]

Il controllo del cursore

Il nuovo comando che vi proponiamo è una semplice routine per il posizionamento del cursore in qualunque parte dello schermo, senza usare dei trucchi di programmazione BASIC come quello che segue:

100 RS= «(premere il tasto di cursore a destra 39 volte)».

110 D\$= «(premere il tasto di cursore in basso 23 volte)».

500 C=22:R=15: REM
MUOVE IL CURSORE
ALLA COLONNA 22, RI-
GA 15

$$510 \text{ M\$} = \text{LEFT\$ (R\$, C)} + \text{LEFT\$ (D\$, R)}$$

520 PRINTMS.

In questo modo (ovviamente le righe 510 e 520 potrebbero essere chiamate come subroutine, aggiungendo un GOSUB alla linea 500 e un RETURN alla linea 520) si ottiene il posizionamento del cursore sopra descritto, ma non è una brutta tiritera?

Ecco invece una ottima soluzione di cui forniamo il listato in Assembler che voi

potete immettere senza problemi. La forma fornita è adatta per un PET con il BASIC 4, con uno schermo piccolo. Le versioni per gli altri PET lavorano esattamente allo stesso modo e qui troverete anche dei DUMPS in esadecimale per i possessori delle differenti versioni.

E vediamo ora i dumps esadecimali per le varie versioni del PET.

Schermo largo, BASIC 4

033A 20 F5 BE 20 D4 C8
E0 18

0342 90 03 4C 00 BF 86 D8
20

```
034A F5 BE 20 D4 C8 E0
28 B0
```

0352 F1 86 C6 20 71 E0 60

Schermo piccolo, BASIC 4
033A 20 F5 BE 20 D4 C8
E0 18

0342 90 03 4C 00 BF 86 D8
20

034A F5 BE 20 D4 C8 E0
28 BO

0352 F1 86 C6 20 7F E0 60
80 colonne BASIC 4

033A 20 F5 BE 20 D4 C8
E0 18

0342 90 03 4C 00 BF 86 E0
20

034A F5 BE 20 D4 C8 EO
50 BO

0352 F1 86 E2 20 5F EO 60

Controllo cursore

033A 20 F5 BE JSR \$BEF5; Controlla se trova una virgola

033D 20 D4 C8JSR \$C8D4;	Prendi un parametro di 1 byte mettilo in x
--------------------------	---

0340 EO 18 CPX £18; Confrontalo con 24 (num. di righe)

0342 90 03 BCC \$0347; Se OK il carry on

0344 4C 00 BF JMP \$BF00; Se no messaggio «Syntax Error»

0347 86 D8	STX \$D8;	Salva il primo parametro in £ D8
------------	-----------	-------------------------------------

0349 20 F5 BE JSR \$BEF5; Controlla per seconda virgola

034C 20 D4 C8 JSR \$C8D4; Prendi il prossimo parametro

034F EO 28 CPX £\$28; Confrontalo con 40 (num. colonne)

0351 BO F1 BCS \$O344; Se maggiore messaggio «Syntax Error»

0353.86 C6	STX \$C6,	Se OK salva il secondo param. in £C6
------------	-----------	---

0335 20 71 E0 JMP \$EO71; Posiziona il cursore

0358 60 RTS

BASIC 3

033A 20 F8 CD 20 78 D6
E0 18

0342 90 03 4C 03 CE D8
20

034A F8 CD 20 78 D6 E0
28 BO

0352 F1 86 C6 20 5D E2 60

Come si utilizza il tutto? Semplice! I dati che voi avete immesso ora risiedono nel Buffer della seconda unità nastro. Per usare il nuovo comando, in qualunque momento, basta inserire questa linea nel vostro programma:

250 SYS 826, 22, 15

Dove 250 è il numero di riga di programma, 22 è la colonna e 15 è la riga dello schermo dove si vuole mettere il cursore. Quando calcolate la posizione assumete che l'origine (0,0) sia in alto a sinistra sullo schermo. È una buona norma non cercare di mettere il cursore sotto la riga 24 perché ciò potrebbe causare uno scrolling accidentale.

Grafica ad alta risoluzione col VIC 20

di Walter Pistarini

Questo articolo spiega come si può ottenere una grafica di 28160 (ventottomilacentosessanta!) punti sul VIC, con una matrice di 176x160 punti (per i VIC con qualche espansione di memoria, per quelli senza espansione la grafica sarà «solo» di 19584 punti, su una matrice di 136x144 punti).

Il manuale del VIC parla di capacità di grafica ad alta risoluzione. C'è anche una sezione che mostra come fare della grafica a 64x64 punti. Sfortunatamente, non è molto chiaro come gira il tutto. Lo scopo di questo articolo è di gettare un po' di luce nella grafica del VIC.

Per capire la grafica ad alta risoluzione bisogna prima capire come lavorano i caratteri programmabili. Il VIC in realtà non ha un modo di lavorare «grafico», ma ha due possibilità che permettono di creare della grafica. La prima e più importante è la possibilità di cambiare il contenuto del puntatore che normalmente punta alla ROM contenente i caratteri (diciamo meglio, le «informazioni» per costruire i caratteri) e farlo puntare in RAM. L'altra possibilità è che la grandezza di carattere può essere cambiata.

Memorie video e caratteri

Prima di spiegare come queste due «opzioni» si combinano per ottenere una buona grafica, dobbiamo rivedere come i caratteri sono solitamente mostrati sullo schermo. Prima di tutto facciamo un breve ripasso su cosa si intende per «memoria video» e cosa si intende per «memoria caratteri» (character generator in inglese).

Per «memoria video» si intende quella parte di memoria del VIC atta a contenere dei «codici» (un codice è lungo un byte) che dicono al VIC quale carattere si vuole vedere in un determinato punto dello schermo. Si potrebbe dire che è la «fotografia» di quello che vedete sullo schermo. C'è una corrispondenza diretta (1 a 1) tra gli indirizzi (o posizioni) nella memoria video e le posizioni sullo schermo. Per fare un esempio: se avete un codice che indica la lettera «A» nella locazione 23 della memoria video, questa lettera «A» vi sarà mostrata nella posizione 23 dello schermo, e cioè come primo carattere della seconda riga, visto che le righe sono lunghe 22 caratteri.

Quindi: Memoria video = foto del video. In un VIC senza espansioni o con l'espansione da 3K, la memoria video comincia all'indirizzo 7680. Se ci sono altre espansioni sarà a 4096 (normalmente).

Per «memoria caratteri» si intende quella parte della memoria del VIC (normalmente su ROM) che contiene le informazioni necessarie per «costruire» tutti i caratteri disponibili sul VIC. Più esattamente diciamo che questa memoria dice al VIC, dato un certo codice (preso nella memoria video), quali punti dello schermo (detti PIXELS) bisogna «accendere» o «spegnere» per mostrare il carattere identificato da quel codice.

Quindi: Memoria caratteri = foto dei caratteri disponibili.

Ogni byte della memoria video è usato come un indice per trovare nella memoria caratteri le appropriate «informazioni». Nel mondo di lavorare normale, un carattere è costituito da 8 righe di 8 punti ciascuna. Con la «opzione» che espande la grandezza dei caratteri abbiamo 16 righe di 8 punti ciascuna.

Il primo «codice carattere» che potete selezionare è 0 (una @), e mettendo 0 nella prima locazione di memoria video dite al VIC prima di tutto che volete vedere il carattere con il codice 0 nella prima posizione dello schermo e poi di andare a vedere nella memoria caratteri i primi 8 bytes, che gli diranno quali punti accendere sul video in quella posizione. Questo perché ogni carattere, come già detto, è fatto di 8 righe di 8 punti ciascuna, per cui il VIC avrà bisogno di 8 bytes di 8 bits ciascuno per sapere quali dei 64 punti (8 righe x 8 punti) che formano un carattere deve accendere o spegnere. Ogni byte della memoria caratteri definisce (con i suoi 8 bits) una riga di punti di un carattere. Se il bit è a 1 vuol dire che il corrispondente punto deve essere acceso, se è da 0 il punto dello schermo deve essere spento.

Provate, su un VIC senza espansioni, a battere:

/ RUN STOP / RESTORE /

e poi:

POKE 36879,62

POKE 7690,0

Il primo POKE scrive a colo-

rare di blu lo schermo, il secondo mette a metà della prima riga il carattere con codice 0. Dovreste vedere una «@». Nella figura 1 potete vedere le relazioni tra la memoria video, la memoria caratteri e lo schermo.

Se fate il POKE dei valori da 0 a 255 nelle prime 256 posizioni della memoria video, tutti i caratteri possibili vi saranno mostrati in ordine.

Quello che voi vedete sullo schermo non sono i 256 caratteri possibili del VIC, ma tutti i bits che sono messi a uno nei 2048 bytes che cominciano all'indirizzo definito come memoria caratteri! (256 indici che identificano i caratteri x 8 righe x 8 punti = 16384 punti = 16384 bits = 2048 bytes x 8 bits).

Dicendo al VIC che la memoria caratteri è localizzata in RAM, si può controllare a programma quali punti «accendere» o «spegnere» sullo schermo. Questa è la maniera in cui si creano i caratteri speciali.

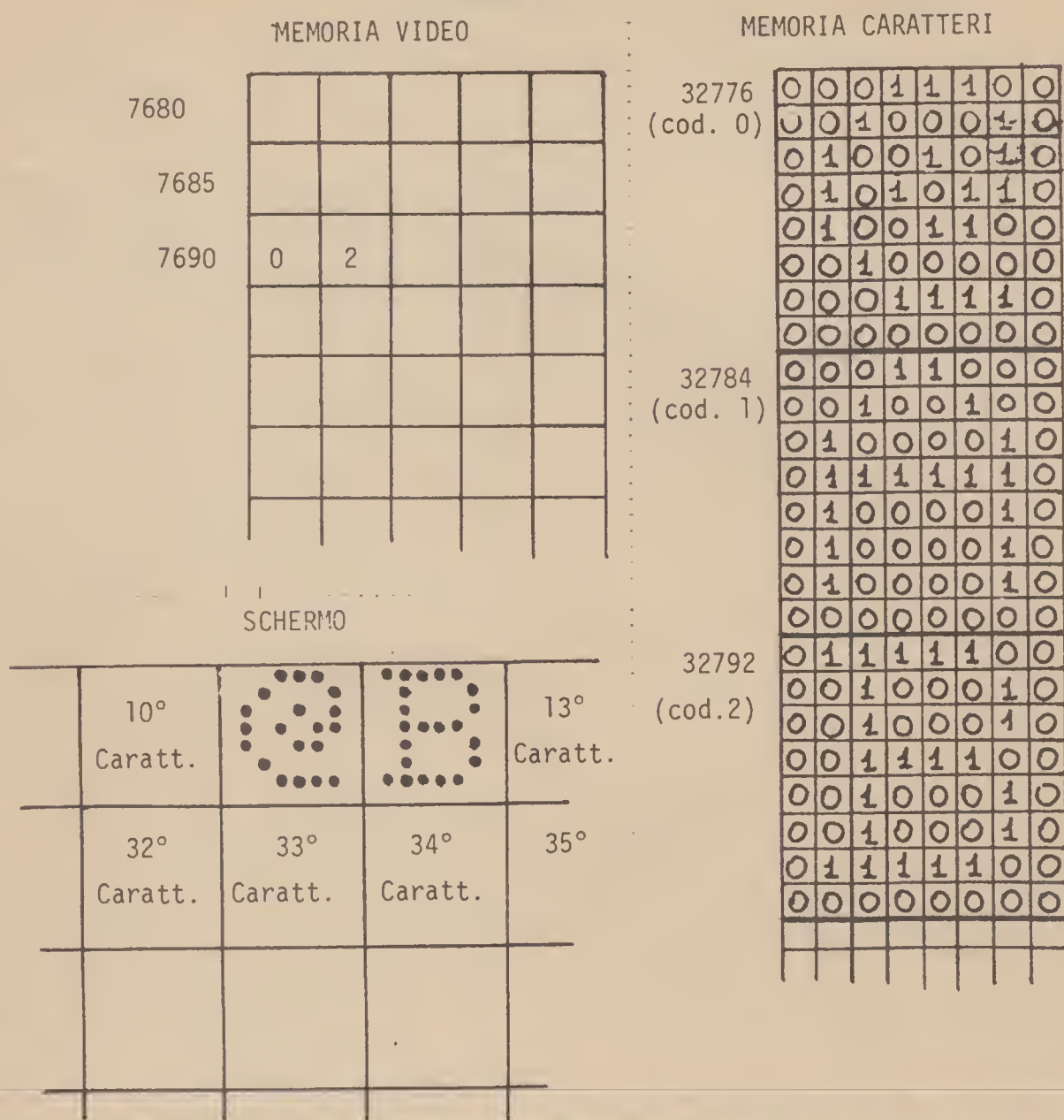
Caratteri e doppia altezza

C'è ancora una importante osservazione da fare. Come tutti sanno, il video del VIC è fatto da 22 caratteri per 23 righe, per un totale di 506 caratteri visibili contemporaneamente. Come si fa a riempire uno schermo di 506 caratteri con solo 256 combinazioni a disposizione (visto che noi ovviamente vogliamo usare la grafica a tutto schermo)? Il problema è che i caratteri sono usati come «indici», per cui se si vuole utilizzare lo schermo in piena grafica avremo bi-

Figura 1

```
10 POKE 36879,62
20 FOR I = 5120 TO 6143 : POKE I,0 : NEXT I
30 POKE 7680,0
40 POKE 36869, (PEEK (36869) AND 240) + 13
50 POKE 5120,1
60 FOR X = 1 TO 10000 : NEXT X
65 POKE 36869,240
70 END
```


Figura 2



sogno di 506 indici.

Il trucco è di usare i caratteri a doppia altezza. Questi caratteri non cambiano la grandezza dei punti sullo schermo, cosicché ogni carattere «copre» realmente il doppio dell'area di un carattere normale. Per cui 256 indici di caratteri a doppia altezza copriranno tanto quanto 512 caratteri normali.

Utilizzo della memoria per la grafica

Per vedere le cose sotto un'altra angolazione, lo schermo del VIC è largo 176

punti (22 caratteri x 8 punti di larghezza ciascuno = 176) e alto 184 (23 righe alte 8 punti ciascuna = 184), per un totale di 32384 punti (4048 bytes). I caratteri di grandezza doppia ci danno 32768 punti (256 indici x 16 righe x 8 punti), cosicché abbiamo risolto tutti i nostri problemi. Giusto? Sbagliato. Il problema è la memoria che il chip del VIC può indirizzare. Come viene spiegato nel modulo di espansione, il chip «VIC» (e cioè l'integrato che «gestisce» il video, all'opposto del Computer VIC), può indirizzare solo memoria da 4096 a 8191

(da c1000 a c1FFF in esadecimale). E in questa memoria ci devono stare sia la memoria video sia la memoria caratteri. Mentre questi 4096 bytes sono sufficienti per contenere un set completo di caratteri a doppia altezza di memoria caratteri, noi abbiamo ancora bisogno di prendere 512 bytes di memoria video da questa stessa area (Ricordate? 22 righe x 23 colonne = 506 caratteri, che arrotondiamo a 512). Diciamo subito, quindi, che non sarà mai possibile avere una grafica ad alta risoluzione di 176x184 punti, ma al massimo otterremo

176x160 punti, dando 3584 bytes alla grafica e 512 alla memoria video.

Gli indirizzi di memoria utilizzabili

Abbiamo discusso finora la maggior parte delle informazioni che voi usate per fare della grafica sul VIC. Ci sono ancora alcuni dettagli tecnici e compromessi concernenti l'ammontare di grafica e di memoria che servono al BASIC. La memoria caratteri può cominciare ad una delle seguenti 4 locazioni di memoria RAM:

Programma 1: VIC con espansione da 3K
Grafica ad alta risoluzione

```

10 PRINT"DIMOSTRAZIONE GRAFICA DEL VIC"
10 PRINT"PC CLUB"
30 POKE1,PEEK(55):POKE 2,PEEK(56) :REM SALVA LA FINE CORRENTE
40 POKE55,0 :REM METTE LA NUOVA FINE A 4096
45 POKE56,4096/256
50 CLR :REM CLR PER AVVERTIRE IL BASIC
100 MC=4096:NR=10 :REM MEMORIA CARATTERI -NUMERO RIGHE
110 NC=22:PV=16 :REM NUMERO CARATTERI - PUNTI VERTICALI
120 PS=NR*PV/2-1 :REM SCELTA FATTORE DI SCALA
200 REM AZZERA LA MEMORIA CHE DIVERRA' MEMORIA CARATTERI
210 FOR X=CH TO MC+NR*NC*PV-1
220 POKE X,0 :NEXT
230 REM TUTTI UNO NELLA MEMORIA NON USATA PER FARE IL BOBDO
240 FOR X=MC+NR*NC*PV TO 7679
250 POKE X,255: NEXT
260 REM SCEGLIE IL FORMATO DEI CARATTERI A 8x16
270 POKE 36867,PEEK(36867) OR 1
280 BY=PEEK(36879) AND 7 :REM LEGGI LO SFONDO CORRENTE
310 FOR X=0 TO NR*NC-1 :REM INDICI NELLA MEMORIA VIDEO
320 POKE 7680+X,X
330 POKE 38400+X,BY
340 NEXT
350 REM RIEMPIE IL RESTO CON UN CARATTERE NON UTILIZZATO
360 FOR X=NR*NC TO 511
370 POKE 7680+X,NR*NC
380 POKE 38400+X,BY
385 NEXT
390 REM DICE AL VIC IL NUMERO DI CARATTERI PER RIGA
400 POKE 36866,(PEEK(36866) AND 128)+NC
410 REM CAMBIA L'INDIRIZZO DELLA MEMORIA CARATTERI
420 POKE 36869,(PEEK(36869) AND 240)+MC/1024+8
450 REM FORMULA DELLA SINUSOIDE
460 FOR X=0 TO NC*8-1
480 Y=INT(SF+SF*SIN(X/10)+1)
490 GOSUB 1000
495 NEXT
510 FOR I=1 TO 10000:NEXT:REM UN PO' PRIMA DI FINIRE
600 POKE55,PEEK(1) :REM RIMETTE A POSTO IL LIMITE MEMORIA
610 POKE56,PEEK(2)
620 SYS(59829) :REM RIMETTE A POSTO IL RESTO
630 END
1000 YR=Y/PV
1010 CA=INT(X/8)+INT(YR)*NC
1020 RI=(YR-INT(YR))*PV
1030 BY=MC+CA*PV+RI
1040 BI=7-(X-(INT(X/8)*8))
1050 POKE BY,PEEK(BY) OR (2+BI)
1060 RETURN

```

4096, 5120, 6144 o 7168 (con un 12, 13, 14 o 15 negli ultimi 4 bits della locazione 36869). La memoria video può essere in una qualunque delle seguenti locazioni di memoria RAM: 4096,

4608, 5120, 5632, 6144, 6656, 7168 o 7680 (con i bits da 4 a 7 della locazione 36869 contenenti 12, 13, 14 o 15, che controllano l'allineamento di 1024 in 1024 e il bit 7 della locazione 36866

che controlla se si è allineati ai 1024 oppure ai 512). Come vedete, la stessa locazione di memoria seleziona sia l'indirizzo della memoria video, sia l'indirizzo della memoria caratteri. Per esse-

re più pratici diciamo che per quanto riguarda la memoria video, la selezione a 4096 viene fatta facendo il POKE di 192 (che sarebbe il valore di un 12 messo nei bits da 4 a 7) nella locazione

Programma 2 - Grafica ad alta risoluzione

```

10 PRINT"GRAFICA CON IL JOYSTICK : SENZA ESPANSIONI"
20 POKE 55,0:POKE 56,20 :REM LIMITA MEMORIA AL BASIC
40 DD=37154:P1=37151:P2=37152
50 X=10:Y=10
60 NC=5120:NR=7 :REM MEMORIA CARATTERI - NUM. RIGHE
65 NC=17:PV=16 :REM NUMERO CARATTERI - PONTI VERTICALI
80 REM AZZERA LA MEMORIA CARATTERI
85 FOR I=NC TO NC+NR*NC*PV-1
90 POKE I,0 :NEXT
100 REM TUTTI UNO NELLA MEMORIA NON USATA PER FARE IL BORDO
110 FOR I=NC+NR*NC*PV TO 7679
120 POKE I,255: NEXT
130 REM SCEGLIE IL FORMATO DEI CARATTERI A 8x16
140 POKE 36867,PEEK(36867) OR 1
150 REM POKE DELLA MEMORIA VIDEO
155 BY=PEEK(36879) AND 7
160 FOR I=0 TO NR*NC-1
170 POKE 7680+I,I
172 POKE 38400+I,BY
175 NEXT
180 REM RIEMPIE IL BESTO CON UN CARATTERE NON UTILIZZATO
190 FOR I=NR*NC TO 511
200 POKE 7680+I,NR*NC
202 POKE 38400+I,BY
205 NEXT
210 REM CAMBIA L'INDIRIZZO DELLA MEMORIA CARATTERI
215 POKE 36866,(PEEK(36866) AND 128)+NC
220 POKE 36869,(PEEK(36869) AND 240)+NC/1024+8
230 REM LOOP PRINCIPALE
240 GOSUB 1010
250 YR=Y/PV
260 CA=INT(X/8)+INT(YR)*NC
270 RI=(YR-INT(YR))*PV
275 BY=NC+CA*PV+RI
280 BI=7-(X-(INT(X/8)*8))
300 POKE BY,PEEK(BY) OR (2^BI)
310 GOTO 240
1000 REM LETTURA JOYSTICK
1010 POKE DD,127:P=PEEK(P2) AND 128
1020 J0=-(P=0)
1030 POKE DD,255:P=PEEK(P1)
1040 J1=-(P AND 8)=0)
1050 J2=-(P AND 16)=0)
1060 J3=-(P AND 4)=0)
1070 IF J0=1 THEN X=X+1
1080 IF J2=1 THEN X=X-1
1090 IF J1=1 THEN Y=Y+1
1100 IF J3=1 THEN Y=Y-1
1110 IF Y>112 THEN Y=112
1115 IF X>136 THEN X=136
1120 RETURN

```

36869, per selezionarla a 5120 il POKE è di 208 (un 13 sempre nei bits 4-7), per 6144 il POKE è di 224 (un 14) e per il 7168 il POKE è

di 240 (un 15). Questo perché normalmente il bit 7 della locazione 36866 indica l'allineamento a 512. Se si volessero i valori 4608,

5632, 6656 e 7680, bisognerebbe fare anche un POKE a 36866 (POKE 36866, (PEEK(36866) OR 128)). Ai valori dei POKE sopra

descritti bisogna aggiungere la selezione della memoria caratteri, semplicemente sommando un 12 (per memoria caratteri a 4096) o un

13 (per 5120) o un 14 (per 6144) o un 15 (per 7168) al valore sopra indicato. Così, se per esempio volessimo la memoria video a 6656 e la memoria caratteri a 5120, dovremmo fare il POKE, sempre a 36869, di 224+13, e cioè: POKE 36869,237.

Una maniera più elegante è quella di fare il POKE del PEEK del valore che c'è, messo in AND con il valore che seleziona la memoria video e sommato con il valore che sceglie la memoria caratteri. In questo modo c'è un piccolissimo rischio (sapete dirmi quale?) ma la leggibilità del programma guadagna enormemente.

Come abbiamo visto, le memorie caratteri e video sono scelte indipendentemente una dall'altra e potrebbero occupare anche la stessa porzione di memoria. Infatti per una grafica con la massima risoluzione possibile, esse devono sovrapporsi per un poco.

Se la memoria caratteri è messa a 4096 (più realisticamente), e la memoria video è messa a 7680 (ancora più realisticamente), (facendo «POKE 36869, (PEEK (36869) AND 240) + 12»), ci sono 3584 bytes disponibili per la grafica (7680 - 4096 = 3584). Ciò ci permette di avere 22 colonne per 10 righe, oppure 176 punti orizzontali per 160 punti verticali (ogni riga è alta 16 punti e larga 8 punti x 22 caratteri il tutto usando i caratteri doppi). Questo però non lascia spazio per programmi BASIC in un VIC senza espansioni di memoria. Se volete spazio per un programma BASIC che occupi non più di 1K, spostate la memoria caratteri a 5120 e mantenete la memoria video a 7680, con il comando:

POKE 36869, (PEEK (36869) AND 240) + 13

Questa configurazione vi dà 2560 bytes per dati di grafica e una griglia di 176 punti (22 caratteri) per 112 (7 caratteri a doppia altezza) usando 2464 bytes. Questa

REMARKS - Programma 1

30 -	Salva in 1 e 2 il limite di memoria per il programma BASIC
40 - 45	Stabilisce un nuovo limite di memoria, che è sotto la memoria caratteri
100 - 120	Mettono a posto le costanti usate in seguito MC è la locazione della memoria caratteri NR è il numero di righe NC è il numero di caratteri per riga PV è il numero di punti verticali x caract. FS è un fattore di scala per centrare la sinusoide Notare che NR*PV è il numero di punti verticali e che NC*8 è il numero di punti orizzontali.
200 - 220	Loop che inizializza la memoria caratteri che useremo per la grafica, mettendo tutto a 0
230 - 250	Inizializza il resto della memoria caratteri mettendo tutti 1
260 - 270	Mette il formato dei caratteri a 8 x 16
280	Legge lo sfondo attuale
310 - 340	Fa il POKE dei numeri da 0 a 219 (153 per i VIC senza espansione) nella memoria video
350 - 385	Fa il POKE di un carattere non usato nel resto della memoria video
390 - 400	Cambia il numero di caratteri per riga (necessario solo per le versioni senza espansioni)
410 - 420	Muove la memoria caratteri nella RAM precedentemente preparata. (lo schermo apparirà restringersi a questo punto)
450 - 495	Disegna la sinusoide, chiamando la subroutine di plot a 1000. Se si vuole utilizzare questo programma per altra grafica, basta mettere la vostra formula per le coordinate X e Y qui mantenendo ovviamente il GOSUB1000
510 -	Aspetta un po' prima di finire
600 - 630	Rimette tutto a posto e chiude
1000 -	Viene fatta una volta solo l'operazione Y/PV (ripetuta in 2 formule)
1010 - 1060	Subroutine per trovare il bit e metterlo a uno con il POKE necessario.

non è una area veramente quadrata su cui disegnare, per cui sarebbe bene cambiare anche il numero di caratteri per riga da 22 a 17:

POKE 36866, (PEEK (36866) AND 128) + 17

Questo ci dà 136 punti orizzontali per 144 punti verticali. Notare, ancora, che tutta questa discussione si applica ad un VIC senza espansione o ad un VIC con una espansione di 3K di memoria.

Esiste una ulteriore complicazione per i VIC con più di 8K. Per questi sistemi, la memoria video parte (normalmente) all'indirizzo 4096, e il programma BASIC comincia a 4608.

Per usare il tipo di grafica di cui stiamo parlando, l'inizio del programma BASIC deve essere spostato sopra l'area usata per le memorie caratteri e video (sopra 8191). Sotto le giuste circostanze, esso può essere fatto dal programma BASIC che sta girando, ma è molto più semplice fare questo spostamen-

to prima di caricare il programma. Vedremo come fare quando descriveremo le modifiche da fare ai programmi se avete questa espansione.

Qualche esempio

Provate a far girare questo breve programma **figura 2** che vi mostrerà alcuni principi fondamentali di questo tipo di grafica.

Guardate cosa è apparso in alto a sinistra sullo schermo. Un punto è stato acceso nella prima riga. La linea 10 del programma colora di blu lo schermo. La linea 20 pulisce tutta la memoria RAM da 5120 a 6143 (diverrà la nuova memoria caratteri). La linea 30 mette un codice carattere di 0 nella memoria video a 7680 (normalmente sarebbe una @). La linea 40 cambia il puntatore alla memoria caratteri da ROM a RAM, indirizzo 5120. La linea 50 crea un nuovo carattere utilizzando solo il primo degli otto bytes che defi-

Figura 3

Numero dei bit
Valori
binari

7 6 5 4 3 2 1 0	
0 0 0 0 0 0 0 1	= 1
0 0 0 0 0 0 1 0	= 2
0 0 0 0 0 1 0 0	= 4
0 0 0 0 1 0 0 0	= 8
1 0 0 0 0 0 0 0	= 128
1 0 0 0 0 0 0 1	= 129
1 1 1 1 1 1 1 1	= 255

niscono il carattere con codice 0, per tutte le altre linee del carattere sono «spente». Questa linea 50 mette a 1 il bit nella posizione 0 (quello più a destra nel byte) per cui vedremo il punto più a destra della prima riga del primo carattere accendersi. La linea 60 fa in modo che il VIC aspetti un po' prima di mostrare «READY» e finire la dimostrazione.

Provate ora a codificare diversi valori nella riga 50, tenendo presente che i valori variano da 0 (nessun punto

Locazioni di memoria utili per la grafica ad alta risoluzione

43,44	Puntatore all'inizio del programma BASIC. Dice al BASIC il numero di pagina di dove comincia il programma. Numero di pagina = Indirizz/256 Per ricavare l'indirizzo bisogna fare: (contenuto locazione 44)x256+(conten. locaz. 43). Per l'espansione di 8K: normalmente 1,18; cambiare a 1,32 per far cominciare il programma a 8192	4096	o Indirizzi possibili per la memoria caratteri
45,46	Puntatore fine programma BASIC, inizio Variabili	5120	o
47,48	Puntatore fine variabili, inizio matrici	6144	o
49,50	Puntatore fine matrici	7168	
51,52	Puntatore fine stringhe (si muovono verso il basso)	8192	Primo indirizzo della espansione da 8K: se si sposta il programma a questo indirizzo, bisogna che il primo byte sia a 0
55,56	Limite della memoria al BASIC	36869	Puntatore alla memoria caratteri e video. Per il VIDEO: POKE di 192 per puntare 4096; 224 per puntare 6144; 208 per puntare 5120; 240 per puntare 7168. Per CARATTERI: sommare al valore del POKE per il video: 12 per puntare 4096; 14 per puntare 6144; 13 per puntare 5120; 15 per puntare 7168.
57,58	Numero della Linea di programma corrente	36866	Per cambiare il numero di caratteri per riga NC = numero caratteri: (PEEK(36866)AND128)+NC Per cambiare allineamento della memoria video ai 512 o ai 1024(Bit 7 a 1 per 512). Normalmente: VIC senza espansione = allineamento a 512; VIC con 3K espansione = allineamento a 512; VIC con 8K espansione = allineamento a 1024.
59,60	Numero della Linea di programma precedente	36867	Per mettere il format dei caratteri a 8x16 (PEEK(36867) OR 1)
643,644	Limite della memoria al BASIC	36879	Colora lo schermo
641,642	Puntatore all'inizio del programma BASIC. Contiene il numero di pagina. Con l'espansione a 8K contiene 0,18; cambiare a 0,32 per 8192	32768	Locazione della memoria caratteri standard
648	Puntatore memoria video, deve essere modificato solo con l'espansione di 8K (metterlo a 30)	38400	Locazioni dei colori (in parallelo con la memoria video)
	Inizio reale del programma BASIC	37888	Locazioni dei colori (espansione 8K)
1024	Con espans. 3K	59829	Routine che rimette a posto gli indirizzi per tornare allo standard: va chiamata col SYS(59829)
4096	Senza espansioni		
4608	Con espans. 8K		
	Inizio MEMORIA VIDEO standard:		
4096	Con espansione da 8K		
7680	Senza espansioni o con espansione da 3K		

acceso) a 255 (tutti i punti accesi).

Per allargare la vostra comprensione, fate le seguenti modifiche al programma precedente e fatelo girare:

```
30 U=0 : FOR J=7680
TO 7701 : POKE J U : U =
U+1 : NEXT
```

50 POKE 5128,1

Vediamo ora come si lavora in grafica con i caratteri a grandezza doppia. Provate a battere:

POKE 36879,62

POKE 36867, (PEEK(36867) OR 1)

Non stupitevi e andate avanti «al buio»:

POKE 7690,0

Per rimettere tutto a posto battere:

POKE 36867, (PEEK(36867) AND 254)

Insieme a della «spazzatura», dovrete vedere due caratteri nel mezzo della prima riga: una «@» sopra una «A». Il secondo POKE ha cambiato la matrice dei caratteri del VIC da 8 linee di 8 punti a 16 linee di 8 punti per ogni carattere. Per cui il VIC questa volta va a leggere i primi 16 bytes della memoria caratteri per costruire il carattere con codice 0. Dato che i secondi 8 bytes identificano una «A» (i primi otto li abbiamo già visti, sono una «@»), noi vediamo entrambi i caratteri. In questa maniera se facciamo il POKE dei valori da 0 a 253 nella memoria video, avremo lo schermo occupato in tutte le sue 506 posizioni.

Il calcolo delle coordinate X e Y

Ora che sappiamo come accendere o spegnere un punto sullo schermo cambiando un bit nella memoria caratteri, dobbiamo fare in modo che il nostro programma prenda delle coordinate X (orizzontale, da 0 a 160) e Y (verticale, da 0 a 176) e le trasformi nel corrispondente indirizzo del byte, dicendoci anche quale bit in quel byte identifica il nostro punto. Le seguenti formule sono tratte dal manuale del VIC:

Carattere

CA =

$\text{INT}(X/8) + \text{INT}(Y/PV) * NC$

Riga del carattere

RI =

$(Y/PV - \text{INT}(Y/PV) * PV)$

Byte

$BY = MC + CA * PV + RI$

Bit

$BI = 7 - (X - (\text{INT}(X/8) * 8))$

Per queste formule, X rappresenta la coordinata orizzontale da sinistra a destra e Y la coordinata verticale dall'alto verso il basso. NC è il numero di caratteri per riga, che vedremo più avanti. PV è il numero di righe di punti per carattere (8 per i caratteri a grandezza normale). MC è l'indirizzo della memoria caratteri.

Per mettere a 1 qualunque bit con le coordinate X e Y, dopo le precedenti formule fare:

POKE

$BY, \text{PEEK}(BY) \text{CR}(2 \uparrow BI)$

L'elevazione a potenza serve a fare in modo di ottenere dei valori binari «interi», e cioè che identifichino un singolo bit. Questo è possibile solo se i valori sono in potenza di due. Il risultato sarà: 0,1,2,4,8,16...ecc.

Notare anche che queste formule partono dal presupposto che si sia precedentemente riempita la memoria video con valori crescenti da 0 al numero di caratteri che avete a disposizione (esempio: 153 per un VIC senza espansioni, 220 per un VIC con espansioni).

Il programma 1

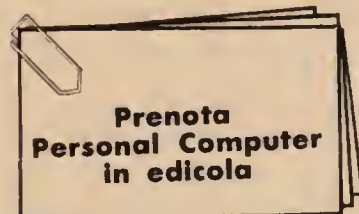
Il primo programma usa la grafica ad alta risoluzione per disegnare una sinusoide. Se voi gli date un'occhiata, troverete che buona parte del programma serve a mettere a posto le cose per la grafica, e che le linee da 460 a 490 sono quelle che creano il disegno da mostrare sul video (una semplice curva sinusoidale). Il programma come listato è per un VIC con una espansione di memoria di 3K. Se avete un VIC senza espansioni, cambiate le seguenti linee e rimuovete tutte le REMs. Ciò vi darà un campo di 136 per 144 punti:

```
45 POKE 56,5120/256
100 MC=5120:NR=7
110 NC=17:PV=16
```

Se invece avete una espansione da 8K dovete rimuovere le righe 40, 45 e 50, dato che la fine della memoria è sopra la memoria video. Dovrete anche battere i seguenti comandi in modo «diretto» (non a programma) prima di caricare il programma BASIC. Queste istruzioni spostano la memoria video dove essa è nella versione standard (senza espansioni) e cambiano l'inizio del programma BASIC in modo tale che sia sopra la memoria video. Questo vi permette di usare la memoria da 4096 a 7680 per le memorie video e caratteri.
POKE 36866,150:POKE 36869,240 :

POKE 648,30 CLR
POKE 43,1 : POKE 44,32
POKE 642,32
POKE 8192,0
CLR : NEW

Il primo POKE mette la selezione dell'allineamento della memoria video a 512 (per poterla poi mettere a 7680), il secondo mette la memoria caratteri a 4096 e quella video a 7168+512, il terzo dice il numero di pagina della memoria video (30x256 = 7680). Voi dovette premere il tasto CLR/HOME per pulire lo schermo dopo aver battuto la prima riga. Ciò dirà al BASIC che avete cambiato la locazione della memoria video. Il POKE a 43,44 e 642 informano il BASIC che deve mettere il programma sopra la memoria video, e cioè a 8192 (32x256 = 8192), mentre il POKE a 8192 mette a 0 la prima locazione di memoria utile per il programma BASIC: è un passo obbligatorio quando si sposta l'inizio del programma.



Realizzare nuovi caratteri

Questa è una procedura che potrete anche inserire come routine nei vostri programmi che vi permetterà di utilizzare le possibilità di alta risoluzione del vostro Vic 20. Basta che abbiate la versione di base dei Vic, cioè i 3.5 Kb di memoria utilizzabile (o «memoria viva»). Tale procedura fa uso della possibilità di usare la «memoria morta» partendo dall'indirizzo N. 36869. Provate prima di scrivere l'intera procedura, a digitare: «POKE 36869, 220 RETURN», i caratteri da questo punto in poi, saranno svincolati e dipenderanno solo dal contenuto della

Listato - Nuovi Caratteri

```
10 REM,VIC 20
20 DIMA(7):N=0:GOTO10000
25 FORX=0TO7:ACX)=0:NEXT
26 I=0:J=0:GOTO 1500
30 POKE 7680+22*I+J,160
35 POKE 36800+22*I+J,6
40 GETA$
50 IFA$=CHR$(13)THEN A(1)=A(1)+2*(7-J)
55 POKE 7680+22*I+J,160
56 J=J+1
60 IFA$=" " THEN POKE 7680+22*I+J,32
65 J=J+1
70 IF J>7 THEN I=I+1:J=0
80 IF I>7 THEN I10
90 POKE 7680+22*I+J,32
100 GOTO30
110 FORX=0TO7
120 POKE 6144+N*8+X,ACX)
130 NEXT
140 INPUT "UN ALTRO CARATTERE":R$
150 IFR$="SI" THEN N=N+1:GOTO20
160 IF R$="NO" THEN I180
170 GOTO140
180 PRINT:POKE 36869,254:N=N+1
190 END
1500 PRINT "          L0
1510 PRINT "          L1
1520 PRINT "          L2
1530 PRINT "          L3
1540 PRINT "          L4
1550 PRINT "          L5
1560 PRINT "          L6
1570 PRINT "          L7
1580 PRINT "[ ] [ ] [ ] [ ] [ ] [ ] [ ]"
1590 PRINT "01234567"
1600 GOTO30
10000 FORX=6144TO7680
10010 POKEX,0:NEXT
10020 GOTO20
```

«memoria viva».

Per tornare ai caratteri da voi abitualmente usati sul vostro Vic, basterà premere Run o Stop o Restore.

Quindi, avendo a disposizione un bel po' di «memoria viva», potremo contare su un numero elevato di caratteri in più, e tutti da noi scelti, creati e voluti.

Come fare per creare dei caratteri nuovi?

Nel caso non fosse a voi noto, un video è un insieme di punti e per far scrivere una parola devo «accendere» alcuni punti, ed altri lasciarli «spenti» come tante piccole lampadine.

O, meglio dire, devo crearmi una «matrice» di punti, sapendo che il mio video è una matrice quadrata di 256 punti per le righe, per altrettanti 256 per le colonne, cioè da 0 a 255.

Ponendo 0 come valore ad un punto (X, Y) avrò che tale punto sarà «spento», mentre con il valore ad 1, avrò che tale punto sarà «acceso». Quindi il carattere viene creato, (forse sarebbe meglio dire solo visualizzato) in base ai punti «accesi».

Ogni carattere è formato da una matrice di 8 righe x 8 colonne.

Simulare l'istruzione PLOT sul C.64

Questo programma, di Paolo Peranzoni di Padova, colma una grossa lacuna del BASIC del C-64 e cioè l'istruzione PLOT/DRAW. È un programma interamente in linguaggio macchina (quindi OK come occupazione di memoria e velocità), è originale ed è ad alto contenuto tecnico. La documentazione è abbastanza completa, l'uso è relativamente facile (una volta capite le regole di «attivazione» basta dare un SYS... e le coordinate) ed è veramente molto veloce. Il programma in linguaggio macchina per COMMO-

DORE 64 ha lo scopo di simulare l'istruzione PLOT, purtroppo assente dal BASIC Commodore, viene usato il comando SYS per chiamare la routine in I.m., per cui, dopo aver posto PL=49292 (vedi programma BASIC dimostrativo, linea 80), la sintassi del comando è:

SYSPL, X, Y, per disegnare il punto coordinate (X, Y), dove X ed Y stanno per qualunque espressione algebrica calcolabile in BASIC oppure

SYSPL, X, Y, X1, Y1,... per disegnare più punti oppure

SYSPL, X, Y TO X1, Y1 per disegnare il segmento che unisce (X, Y) con (X1, Y1) oppure

SYSPL, X, Y TO X1, Y1 TO X2, Y2 TO... per disegnare più segmenti consecutivi.

Il comando **SYSPL TO X, Y...** traccia un segmento dall'ultimo punto precedentemente disegnato fino ad (X Y); se non è ancora stato disegnato nessun punto, oppure se è stato azzerato il flag di abilitazione tracciamento segmenti (locaz. \$FE), il comando equivale a **SYSPL, X, Y...**

Si possono liberamente mescolare le due modalità (singoli punti e segmenti) in una stessa istruzione, purché ovviamente non si ecceda la lunghezza di 80 caratteri di una linea BASIC.

Prima di poter usare l'istruzione **SYSPL** è necessario:

1) Caricare la routine in I.m. in memoria: ciò è realizzato

dal programma BASIC mediante la lettura di una serie di DATA decimali.

2) Pulire la pagina grafica prescelta e riempire la mappa colore; ciò è effettuato da una routine in I.m., che viene chiamata dal BASIC mediante il comando **SYSCL** (vedi listato, linee 70, 110 e 140).

3) Predisporre il chip VIC II per la grafica: ciò è ottenuto con le linee 80 e 90 del programma BASIC: non ho implementato questa routine in I.m. dato che non ci sono problemi di velocità.

Il programma in I.m. fa largo uso di routines del BASIC; per individuare i punti di ingresso di tali routines mi sono servito del libro **VIC REVEALED** di Nick Hampshire, dato che il BASIC dei due computer è uguale, a parte le zone di memoria; purtroppo il libro è ricco di inesattezze che rendono faticoso un corretto uso delle routine elencate. (W.P.)

Note e commenti ai programmi - Istruzione PLOT

Il programma in I.m. si sviluppa secondo il diagramma di flusso in figura 1 nel quale X, Y indicano le coordinate del punto da disegnare (o dell'inizio del segmento nel caso di **SYSPL TO**) e X1, Y1 le coordinate della fine del segmento; N rappresenta il numero di passi necessari per il tracciamento, mentre ΔX e ΔY rappresentano gli incrementi delle due variabili ad ogni passo.

Le diverse quantità da memorizzare vengono sistemate in blocchi da 5 bytes ciascuno, denominati ACC3, ACC4, ecc., in analogia con ACC1 e ACC2 (primo e secondo float. acc.). Gli indirizzi di partenza e i contenuti sono:

ACC3 da \$00 57	contiene N prima della trasf. in intero
ACC4 da \$00 5C	contiene ΔY e poi $\Delta Y'$
ACC5 da \$0 2A7	contiene X dell'ultimo punto
ACC6 da \$0 2AC	contiene Y
ACC7 da \$0 2B1	contiene ΔX e poi $\Delta X'$

Per il contatore dei passi si utilizzano poi le locazioni \$4B e \$4C; altre locazioni che non interferiscono col BASIC sono usate per passare parametri.

La locazione \$0 2 è usata per il codice colore (semi-byte alto = colore punto, semi-byte basso = colore sfondo).

I vari arrotondamenti all'intero si rendono necessari sia per garantire una corretta rappresentazione cartesiana (per inciso: l'origine è in basso a sinistra, e gli assi orientati come di consueto), sia per evitare irregolarità nel tracciamento dei segmenti.

Il diagramma di flusso della subroutine **PLOT** è mostrato in figura 2.

L'algoritmo usato riprende noti metodi proposti, in BASIC, anche dal manuale **COM-MODORE**, con qualche miglioramento legato all'uso delle funzioni logiche. Il puntatore alla memoria di schermo è costituito dalle locazioni \$22 e \$23.

Il programma BASIC oltre a caricare la routine in I.m., costituisce una dimostrazione dell'uso di questa istruzione; disegna una serie di quadrati concentrici, e poi una figura di Lissajus. La scelta del colore (codice 20 7 = punti grigio-medio su sfondo grigio-chiaro) è legata all'uso di un monitor monocromatico a fosfori verdi, dove offre risultati soddisfacenti senza dover modificare luminosità e contrasto rispetto alla pagina testo; su un televisore a colori si possono usare altre combinazioni, tenendo presente però che spesso insorgono aberrazioni cromatiche.

Ultima annotazione: la somma del DATA (utile per un controllo di esattezza) deve dare 50830.

Push Over per VIC 20

Il listato di «Push Over», gioco di strategia di scacchiera viene proposto dal fiorentino **Simone Marinai**. «Push Over» è un gioco da scacchiera (5x5), che richiede di allineare quattro pedine del proprio colore in orizzontale o verticale (non diagonale), per vincere. La differenza con i normali «filetti» sta tutta nel fatto che è vietato deporre pedine direttamente sulla scacchiera, ma bisogna al contrario appoggiarle all'esterno di essa, intorno, facendo poi scorrere la fila di pedine fino a far entrare la propria (da qui il nome del gioco). Esempi e figure sono stati discussi su **Personal Computer** n. 5/83, ma il listato del nostro amico fiorentino dovrebbe aiutarvi e chiarire ogni dubbio. Buon lavoro! (L.F.)

(Il listato è a pag. 36)

Fig. 1 - Diagramma di flusso del programma L.M. disassemblato

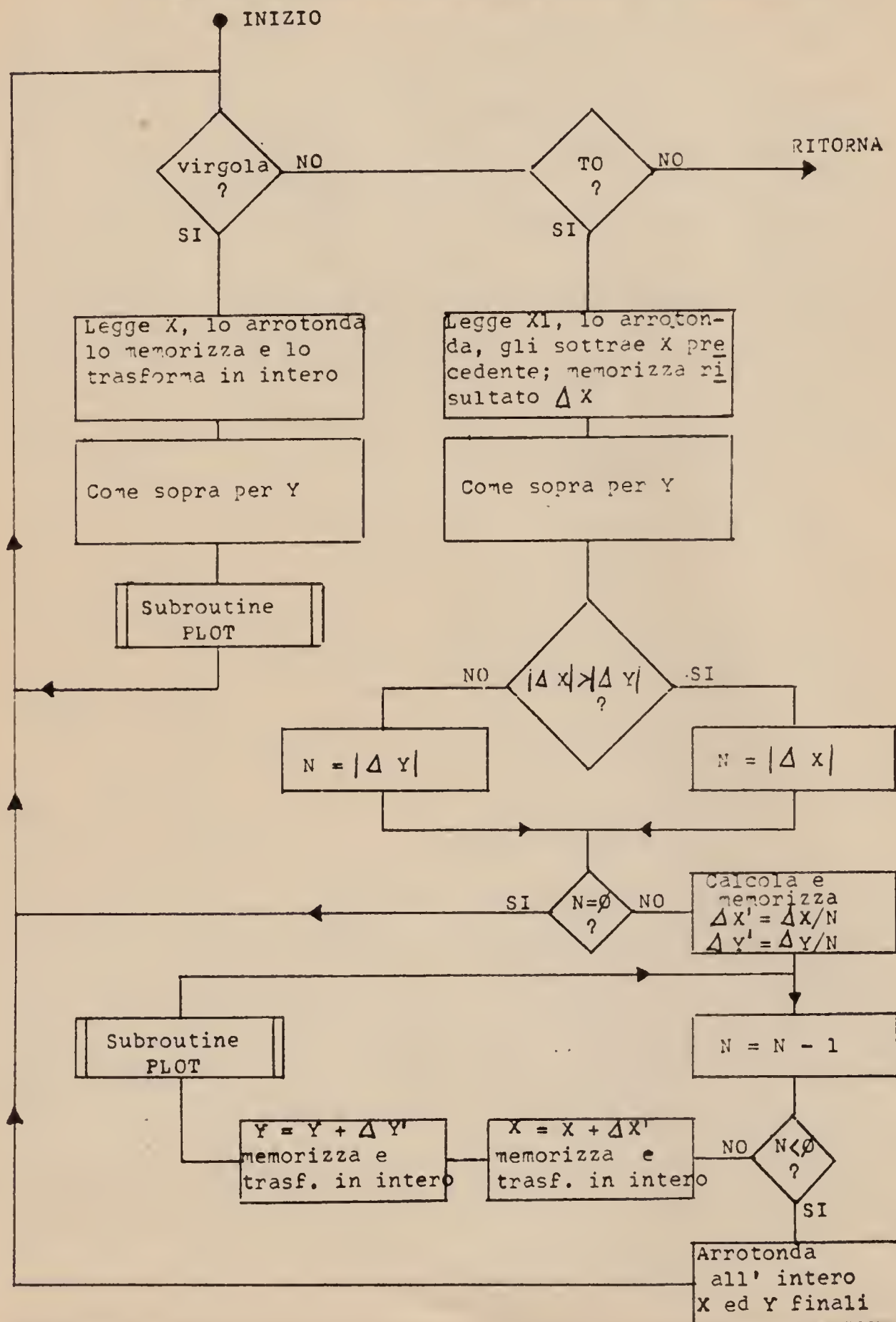
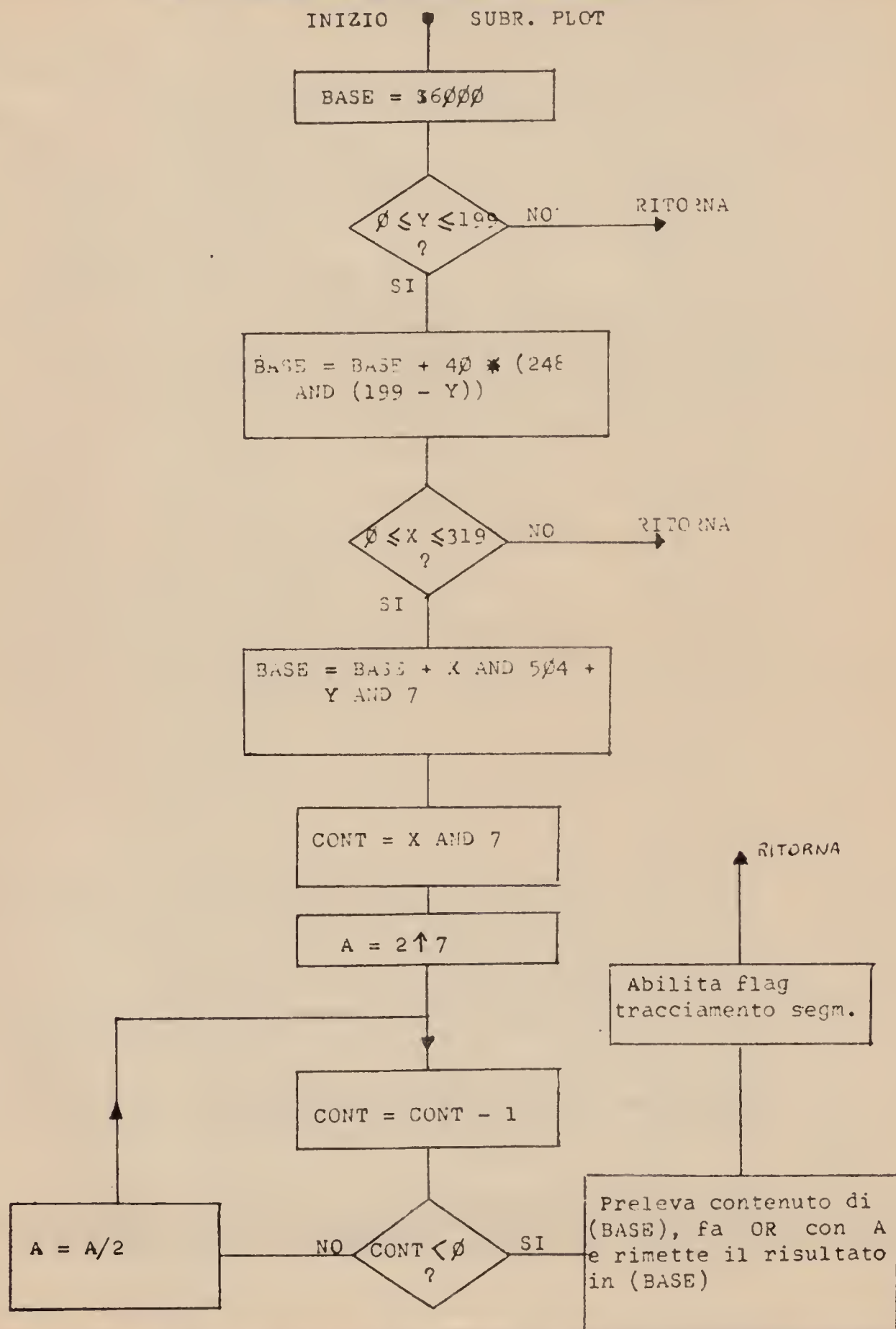


Fig. 2 - Diagramma di flusso della subroutine PLOT.



Programma in L.M. disassemblato - Istruzione PLOT

C000	PLOT	LDX H3AC	
C002		LDY H302	
C004		JSR 3BBD4	;trasf. ACC1 in ACC6
C007		JSR 3B849	;somma 0.5 ad ACC1
C00A		JSR 3B1BF	;trasf. fl. point in intero
C00D		LDA H300	;
C00F		STA 322	;
C011		LDA H360	;
C013		STA 323	;puntatore origine schermo grafico
C015		LDA 364	;byte alto coord. Y
C017		BNE 3C064	;se ≠0 va a RETURN
C019		LDA H3C7	;decimale 199
C01B		SEC	
C01C		SBC 365	;inverte coord. Y
C01E		BCC 3C064	;se negativo va a RETURN
C020		TAY	;salva A in Y
C021		AND H3F8	;maschera i tre bit più bassi
C023		STA 365	
C025		LSR	;la routine che segue, fino a 3C039,
C026		ROR 322	; moltiplica l'accumulatore per 40
C028		ROR	; ed aggiunge il risultato al puntatore
C029		ROR 322	; dello schermo grafico; è scritta per
C02B		ADC 365	; esteso, senza fare uso di cicli, per
C02D		ROR	; renderla più veloce
C02E		ROR 322	;
C030		ROR	;
C031		ROR 322	;
C033		ROR	;
C034		ROR 322	;
C036		ADC 323	;
C038		STA 323	; -----
C03A		LDA 315	;byte alto coord. X
C03C		LSR	
C03D		BNE 3C064	;se maggiore di 1 va a RETURN
C03F		LDA 314	;byte basso coord. X
C041		BCC 3C049	;se X minore di 256 non incrementa 323
C043		INC 323	
C045		CMP H340	;confr. X con 320(dec.)
C047		BCS 3C064	;se maggiore o uguale va a RETURN
C049	NOINC	AND H3F8	;maschera i tre bit più bassi
C04B		STA 365	
C04D		TYA	;richiama Y in A
C04E		AND H307	;maschera i cinque bit più alti
C050		ADC 365	
C052		TAY	;trasf. A in Y (offset per ind. indiretto)
C053		LDA 314	;byte basso X

C055		AND H307	;maschera i cinque bit più alti
C057		TAX	;routine per settare il bit corrispon
C058		LDA H380	; dente al punto da accendere (fino
C05A	LOOP	DEX	; a RETURN escluso)
C05B		BMI 3C060	;
C05D		LSR	;
C05E		BNE 3C05A	;
C060	LPEND	ORA (322),Y	;
C062		STA (322),Y	; -----
C064	RETURN	LDX H3FF	
C066	RET2	STX 3FE	;abilita tracciamento segmenti
C068		CLC	
C069	RET1	RTS	;ritorna alla routine chiamante
C06A	CLEAR	LDA H300	;questa routine (fino a 3C08B) pulisce
C06C		LDX H320	; lo schermo grafico e predispone la
C06E		LDY H360	; mappa colore
C070		CLC	;
C071	BIS	STY 315	;
C073		LDY H300	;
C075		STY 314	;
C077	LOCP1	STA (314),Y	;
C079		INY	;
C07A		BNE 3C077	;
C07C		INC 315	;
C07E		DEX	;
C07F		BNE 3C077	;
C081		BCS 3C066	;
C083		LDA 302	;colori sfondo e punto
C085		LDX H304	;
C087		LDY H35C	;
C089		SEC	;
C08A		BCS 3C071	;-----
C08C	START	LDY H300	;punto di ingresso routine
C08E		LDA (37A),Y	;analizza carattere seguente SYSFL
C090		CMP H32C	;carattere 'virgola'
C092		BNE 3C0BF	;se non è 'virgola' va a NOCOM
C094	SINGLE	JSR 30073	;legge un carattere
C097		JSR 3AD8A	;legge coord. X
C09A		JSR 3B849	;aggiunge .5 per arrotondare
C09D		JSR 3BCCC	;prende la parte intera
C0A0		LDX H3A7	;
C0A2		LDY H302	;
C0A4		JSR 3BBD4	;trasferisce ACC1 in ACC5
C0A7		JSR 3B1AA	;trasforma fl.point in intero
C0AA		STY 314	
C0AC		STA 315	
C0AE		JSR 3AEFD	;verifica virgola
C0B1		JSR 3AD8A	;legge coord. Y

Programma in L.M. disassemblato - Istruzione PLOT

C0B4	JSR 3B849	;aggiunge .5
C0B7	JSR 3BCCC	;parte intera
C0BA	JSR 3C000	;va alla subr. PLOT
C0BD	BCC 3C08C	;torna a START
C0BF	NOCOM CMP H3A4	;codice BASIC comando TO
C0C1	BNE 3C069	;se non è 'TO' va a RET1 (ritorna)
C0C3	LDA 3FE	;flag abilitazione tracciamento segm.
C0C5	BEQ 3C094	;se =0 va a SINGLE (ignora TO)
C0C7	JSR 30073	;legge un carattere
C0CA	JSR 3AD8A	;legge coord. X
C0CD	JSR 3B849	;aggiunge .5
C0D0	JSR 3BCCC	;parte intera
C0D3	LDA H3A7	
C0D5	LDY H302	
C0D7	JSR 3B850	;ACC5-ACC1 in ACC1
C0DA	LDX 3B1	
C0DC	LDY 302	
C0DE	JSR 3BBD4	;trasferisce ACC1 in ACC7
C0E1	JSR 3AEFD	;verifica virgola
C0E4	JSR 3AD8A	;legge coord. Y
C0E7	JSR 3B849	;+.5
C0EA	JSR 3BCCC	;parte intera
C0ED	LDA H3AC	
C0EF	LDY H302	
C0F1	JSR 3B850	;ACC6-ACC1 in ACC1
C0F4	JSR 3BBC7	;ACC1 in ACC4
C0F7	LSR 366	;valore assoluto di ACC1
C0F9	LDA H3B1	
C0FB	LDY H302	
C0FD	JSR 3BA8C	;trasferisce ACC7 in ACC2
C100	LDA 36A	;primo byte mantissa ACC2
C102	AND H37F	;maschera il bit più alto
C104	STA 36A	
C106	LDA H369	
C108	LDY H300	
C10A	JSR 3BC5B	;confronta ACC1 con ACC2
C10D	ASL	
C10E	BCC 3C117	;se ACC1 è maggiore di ACC2 va a MAG
C110	LDA H369	
C112	LDY H300	
C114	JSR 3BBA2	;trasferisce ACC2 in ACC1
C117	MAG JSR 3B1AA	;trasforma fl.point in intero
C11A	STY 34B	;contatore passi tracciamento
C11C	STA 34C	;
C11E	BNE 3C126	;se maggiore di 0 va a DRAW
C120	TAX	
C121	BNE 3C126	;

C123		JMP 3C08C	;torna a START
C126	DRAW	JSR 3B391	;trasforma intero in fl. point
C129		JSR 3BBCA	;trasferisce ACC1 in ACC3
C12C		LDA H3B1	
C12E		LDY H302	
C130		JSR 3BB0F	;ACC7/ACC1 in ACC1
C133		LDX H3B1	
C135		LDY H302	
C137		JSR 3BBD4	;trasferisce ACC1 in ACC7
C13A		LDA H357	
C13C		LDY H300	
C13E		JSR 3BBA2	;trasferisce ACC3 in ACC1
C141		LDA H35C	
C143		LDY H00	
C145		JSR 3BB0F	;ACC4/ACC1 in ACC1
C148		JSR 3BBC7	;ACC1 in ACC4
C14B	LOOP2	LDA 34B	;
C14D		BNE 3C153	;
C14F		DEC 34C	;decrementa contatore passi (byte alto)
C151		BMI 3C187	;se il tracciamento è finito va a START1
C153		DEC 34B	;decrementa contatore passi (byte basso)
C155		LDA H3B1	
C157		LDY H302	
C159		JSR 3BBA2	;trasferisce ACC7 in ACC1
C15C		LDA H3A7	
C15E		LDY H302	
C160		JSR 3B850	;ACC5-ACC1 in ACC1
C163		LDX H3A7	
C165		LDY H302	
C167		JSR 3BBD4	;trasferisce ACC1 in ACC5
C16A		JSR 3B849	;+.5
C16D		JSR 3B1AA	;trasforma fl.point in intero
C170		STY 314	
C172		STA 315	
C174		LDA H35C	
C176		LDY H300	
C178		JSR 3BBA2	;trasferisce ACC4 in ACC1
C17B		LDA H3AC	
C17D		LDY H302	
C17F		JSR 3B850	;ACC6-ACC1 in ACC1
C182		JSR 3C000	;va alla subr. PLOT
C185		BCC 3C14B	;va a LOOP2
C187	START1	LDA H3A7	;questa routine (fino alla fine) arrotonda
C189		LDY H302	; all'intero il contenuto di ACC5 e di
C18B		JSR 3BBA2	; ACC6 , che contengono le coordinate
C18E		JSR 3B849	; X ed Y dell'ultimo punto disegnato
C191		JSR 3BCCC	;
C194		LDX H3A7	;
C196		LDY H302	;

Programma in L.M. disassemblato - Istruzione PLOT

C198	JSR 3BBD4	;
C19B	LDA #3AC	;
C19D	LDY #302	;
C19F	JSR 3BBA2	;
C1A2	JSR 3B849	;
C1A5	JSR 3BCCC	;
C1A8	LDX #3AC	;
C1AA	LDY #302	;
C1AC	JSR 3BBA2	;
C1AF	JMP 3C08C	;----- FINE ROUTINE L.M. -----

Listato programma Basic - Istruzione PLOT

```

0 REM ----- PLOT -----
1 DATA 162,172,160,2,32,212,187,32,73,184,32,191,177,169,0,133,
  34,169,96,133,35
3 DATA 165,100,208,75,169,199,56,229,101,144,68,168,41,248
5 DATA 133,101,74,102,34,106,102,34,101,101,106,102,34,106,102,
  34,106,102,34
7 DATA 101,35,133,35,165,21,74,208,37,165,20,144,6,230,35
9 DATA 201,64,176,27,41,248,133,101,152,41,7,101,101,168,165,20
11 DATA 41,7,170,169,128,202,48,3,74,208,250,17,34,145,34,162,
  255,134,254,24,96
12 REM -----CLEAR-----
13 DATA 169,0,162,32,160,96,24,132,21,160,0,132,20,145,20,200,
  208,251
15 DATA 230,21,202,208,246,176,227,165,2,162,4,160,92,56,176,229
16 REM
17 REM ----- START -----
18 REM
19 DATA 160,0,177,122,201,44,208,43,32,115,0,32,138,173,32,73,184,
  32,204,188
21 DATA 162,167,160,2,32,212,187,32,170,177,132,20,133,21,32,253,
  174,32,138,173
23 DATA 32,73,184,32,204,188,32,0,192,144,205,201,164,208,166,165,
  254,240,205
25 DATA 32,115,0,32,138,173,32,73,184,32,204,188,169,167,160,2,
  32,80,184,162,177
27 DATA 160,2,32,212,187,32,253,174,32,138,173,32,73,184,32,204,
  188,169,172,160
29 DATA 2,32,80,184,32,199,187,70,102,169,177,160,2,32,140,186,
  165,106,41,127
31 DATA 133,106,169,105,160,0,32,91,188,10,144,7,169,105,160,0,
  32,162,187
32 REM ----- MAG -----

```

Listato programma Basic - Istruzione PLOT

```

33 DATA 32,170,177,132,75,133,76,208,6,170,208,3,76,140,192,32,
    145,179,32,202,187
35 DATA 169,177,160,2,32,15,187,162,177,160,2,32,212,187,169,87,
    160,0,32,162,187
37 DATA 169,92,160,0,32,15,187,32,199,187,165,75,208,4,198,76,48,
    52,198,75
39 DATA 169,177,160,2,32,162,187,169,167,160,2,32,80,184,162,167,
    160,2
41 DATA 32,212,187,32,73,184,32,170,177,132,20,133,21,169,92,160,0
43 DATA 32,162,187,169,172,160,2,32,80,184,32,0,192,144,196
45 DATA 169,167,160,2,32,162,187,32,73,184,32,204,188,162,167,
    160,2,32,212,187
47 DATA 169,172,160,2,32,162,187,32,73,184,32,204,188,162,172,
    160,2,32,212,187
49 DATA 76,140,192
50 REM ----- FINE ROUTINE L.M. -----
51 REM
52 REM
59 REM ----- CARICA ROUTINE L.M. -----
60 FOR I=49152 TO 49585: READ P: POKE I,P:NEXT
69 REM ----- DEFINISCE CHIAMATE SUBR. E PULISCE SCHERMO GRAFICO ----
70 PL=49292: CL=49258: SYSCL
79 REM ----- INIZIALIZZA VIC II -----
80 VI=53248: M=VI+17: AD=VI+24: CI=56576
90 POKE M,PEEK(M) OR 32: POKE AD,7*16+8: POKE CI, PEEK(CI) AND 254
99 REM ----- SCELTA COLORI -----
100 POKE 2,207
109 REM ----- DISEGNA QUADRATI -----
110 SYSCL: FOR I=8 TO 80 STEP 8
120 SYSPL,160+I,100+I TO 160+I,100-I TO 160-I,100-I TO 160-I,100+I
    TO 160+I,100+I: NEXT
130 FOR I=0 TO 1000: NEXT
139 REM ----- DISEGNA FIGURE LISSAJUS -----
140 SYSCL: FOR I=0 TO 189: SYSPL TO 160+150*SIN(I/6),100+90*COS(I/10):
    NEXT
150 FOR I=0 TO 1000: NEXT
159 REM ----- RIPRISTINA SCHERMO TESTO -----
160 POKE M,PEEK(M) AND 223: POKE AD,21: POKE CI,PEEK(CI) OR 1
170 PRINT CHR$(147) TAB(207)"PER RIPETERE PREMI UN TASTO"
180 PRINT TAB(210)"PER USCIRE PREMI STOP"
190 GET K$: IF K$="" GOTO 190
200 RUN 70

```


Listato - Push Over

```

260 A1=A(A,B)
270 IF A1=0 THEN PRINT " ": GOTO 300
280 IF A1=1 THEN PRINT "o": GOTO 300
290 PRINT "o";
300 T=T+A1
310 NEXT : NEXT
320 A=A-1 : B=B-1 : GOSUB 550
345 REM SOMME DELLE LINEE PERPENDICOLARI ALLA FOSSA
350 T=0 : IF H=1 THEN 400
360 H=1 : FOR A=0 TO 4
370 FOR B=0 TO 4
380 T=T+A(A,B) : NEXT B=B-1 : GOSUB 550
390 T=0 : NEXT : GOTO 450
400 U=2 : FOR B=0 TO 4
410 FOR A=0 TO 4
420 T=T+A(A,B) : NEXT : U=U-1 : GOSUB 550
430 T=0 : NEXT
445 REM SOMME DELLE LINEE DIAGONALI
450 T=0 : FOR A=0 TO 4 : T=T+A(A,A) : NEXT : U=3 : GOSUB 550
460 T=0 : FOR A=0 TO 3 : T=T+A(A,A+1) : NEXT : GOSUB 550
470 T=0 : FOR A=1 TO 4 : T=T+A(A,A-1) : NEXT : GOSUB 550
480 T=0 : FOR A=0 TO 4 : T=T+A(A,A+4) : NEXT : U=4 : GOSUB 550
490 T=0 : FOR A=0 TO 3 : T=T+A(A,3-A) : NEXT : GOSUB 550
500 T=0 : FOR A=1 TO 4 : T=T+A(A,5-A) : NEXT : GOSUB 550
510 GOTO 110
545 REM SUBROUTINE VERIFICA DELLA SOMMA
550 IF T>35 OR T=4 OR T=5 OR T=13 THEN 570
560 RETURN
570 ON H GOTO 580,600,620,640
580 IF A(1,1)=A(1,2) AND A(1,2)=A(1,3) THEN 650
590 RETURN
600 IF A(1,3)=A(2,3) AND A(2,3)=A(3,3) THEN 750
610 RETURN
620 IF A(1,1)=A(2,2) AND A(2,2)=A(3,3) THEN 750
630 RETURN
640 IF A(1,3)=A(2,2) AND A(2,2)=A(3,1) THEN 750
650 RETURN
660 REM VERIFICA SOMMA
700 IF T>35 OR T=4 OR T=5 OR T=13 THEN 750
710 RETURN
745 REM MESSAGGIO FINALE
750 IF T>35 THEN J$=G2$: GOTO 770
760 J$=G1$
770 GOSUB 780 : PRINT J$ : " HAI VINTO !! " : END
780 PRINT "(home)": SPC(169);"(40 spazi)(home)":SPC(169);: RETURN

```

Vi proponiamo due interessanti programmi, realizzati da Massimo Roberto Garibaldi, per il Commodore 64. Il primo è una Utility che serve a fare copie di blocchi di dischetti, sia che si possieda un solo drive sia che se ne possieda due. È possibile anche formattare i dischetti prima della copia, come è possibile anche fornire la traccia da cui si intende partire (vi ricordo che un dischetto contiene 35 tracce, e la traccia 1 è la più esterna). Il secondo programma inviato da Massimo è in linguaggio macchi-

(non ci sono REMarks, né liste delle variabili, né commenti ai listati). Penso di avervi annoiato abbastanza, per cui passo senz'altro la parola al nostro amico.

Il Programma è particolar-

Il Programma è infatti basato sulla gestione di una parte di memoria che viene caricata dal disco originale e scaricata su quello di backup.

A=USR(1)	LEGGE IL JOYSTICK DEL PORT NO.1
A=USR(2)	LEGGE IL JOYSTICK DEL PORT NO.2
A=USR(3)	LEGGE LA COORDINATA X DELLA LIGHT PEN
A=USR(4)	LEGGE LA LA COORDINATA Y DELLA LIGHT PEN
A=USR(5)	LEGGE LA PADDLE X SUL PORT 1
A=USR(6)	LEGGE LA PADDLE X SUL PORT 2
A=USR(7)	LEGGE LA PADDLE Y SUL PORT 1
A=USR(8)	LEGGE LA PADDLE Y SUL PORT 2

Utility e linguaggio macchina

```

5 PRINT "OK"
FORP=1 TO 1
  PRINT " "
  NEXT
  PRINT "GAME CONTROL READ "
  FORP=1 TO 51
    PRINT " "
    NEXT
    PRINT " "
10 PRINT "ATTENDERE. PREC"
20 E=3247
FORV=E-162 TO E
  READP
  POKEN,A
  NEXT
  SE=140+B
  SW=INT(S/256)
  SL=5-SW*256
40 POKETAS,SL
POKETAS,BH
SP=
100 PRINT "JOYSTICK -A- JOYSTICK -B-E
  X=1
  GOSUB200
110 PRINT "LIGHT PEN -X- LIGHT PEN -Y-E
  X=3
  GOSUB200
120 PRINT "X PADDLE -A- X PADDLE -B-E
  X=5
  GOSUB200
130 PRINT "Y PADDLE -A- Y PADDLE -B-E
  X=7
  GOSUB200
140 PRINT " "
  GOTO120
200 V=JST(X)
Z=IER(V*1)
AS=STR$(V)
ZS=STR$(Z)
210 AS=AS+LEFT$(SPS,6)-LEN(AS))
ZS=ZS+LEFT$(SPS,6)-LEN(ZS))
220 PRINTTAB(6)ASTAR(AS)AS
  RE=64
5000 DATA:162,182,187,219,212,225,199
5010 DATA:59,228,133,106,169,2,133,185
5020 DATA:169,212,133,106,169,25,133,187
5030 DATA:169,209,133,110,169,19,133,189
5040 DATA:169,207,133,212,133,254,169,94
5050 DATA:133,253,32,170,177,162,1,196
5060 DATA:121,240,0,192,0,240,11,200
5070 DATA:76,142,207,177,253,133,251,120
5080 DATA:251,2,94,160,0,76,169,207
5090 DATA:169,1,177,185,73,255,41,91
5100 DATA:169,163,0,32,143,179,94,160
5110 DATA:76,189,207,140,1,177,189
5120 DATA:169,140,0,32,143,179,94,160
5130 DATA:162,64,76,223,207,160,1
5140 DATA:162,64,76,223,207,160,1
5150 DATA:120,76,223,207,160,1,162,120
5160 DATA:120,173,2,220,133,2,169,192
5170 DATA:141,2,220,142,0,220,234,234
5180 DATA:234,234,177,187,169,169,0,32
5190 DATA:143,179,165,2,141,2,220,88
5200 DATA:

```

Dopo, il RUN viene chiesto di quali unità si dispone. Se abbiamo un'unica unità, basterà confermare le risposte che sono presenti sullo schermo come default, se invece vogliamo che la copia avvenga tra diversi drives dovremo modificare i valori proposti e farli accettare con RETURN. Tutto questo permette l'utilizzo del programma anche su Dual Drive.

Verrà poi richiesto se utilizzare o meno la routine di formattazione. Prima che le operazioni di copia inizino dobbiamo ancora fornire al computer la traccia dalla quale vogliamo iniziare. Questo permette un notevole risparmio di tempo in quanto si copieranno solo le tracce che effettivamente contengono dei dati.

Inizia quindi la copia con le richieste di scambio tra il disco originale e di backup.

Il programma è completo di routine per il controllo di errori disco che vengono richiamate ad ogni suo accesso e di continui controlli sull'intervento dell'operatore le cui risposte sbagliate non vengono accettate. Tutto ciò porta ad una estrema facilità d'uso.

È da notare l'uso dell'istruzione WAIT alle linee 9, 55 e 1010 dove il programma resta bloccato finché non viene premuto un carattere. La seguente GET A\$, ripulisce la tastiera da caratteri che altrimenti sporcherrebbero il video.

Game Read

Ho presentato il programma con due listati.

Il primo è in Basic e carica in memoria le routine in L.M. di lettura, l'altro è il disassemblato in mnemonico 6502 delle stesse.

Utility e linguaggio macchina

```

8 PRINT "OK"
DIMAX(255,60)
POKE53200,1
POKE53201,1
1 PRINT "COPY BLOCK CONTENTS"
2 PRINT " "
3 INPUT "COPY FROM UNIT 8100",UF
GOSUB300
DF=VAL(DF)
4 INPUT "COPY TO UNIT 8100",UT
GOSUB300
DT=VAL(DT)
5 PRINT "FORMAT NEW DISK"
PRINT "(V/N) ? "
6 WAIT197,64,255
GETA$
PRINTA$
IFAS="Y" THEN GOSUB1000
7 INPUT "COPY FROM TRACK 10000",T
IFT<10000 THEN PRINT "T",
  GOTO7
8 FORC=1 TO 10
  PRINT "INSERT MASTER DISK"
  PRINT "AND PRESS A KEY"
  WAIT197,64,255
  GETA$
  PRINT "WAIT A MOMENT"
  LF=15
  OPENLF,UF,15,"I" + STR$(DF)
  OPEN5,UF,5,"I"
  GOSUB300
  AA=T
  BB=S
  FORX=0 TO 60
    GOSUB500
    20 PRINT " "
    PRINT "TRACK "T" SECTOR "S" T"
    PRINTLF,"B-P "S,B
    PRINTLF,"U "S,DF,T,S
    GOSUB300
    30 FORL=0 TO 255
      DET=5,AS
      IFAS=" " THEN AA=0
      GOTO40
    35 A=ASC(AS)
    AX(L,X)=A
    40 NEXTL
    NEXTX
    CLOSELF
    CLOSE5
    PRINT "INSERT BACKUP DISK"
    PRINT "AND PRESS A KEY"
    T=AA
    S=BB
    55 WAIT197,64,255
    GETA$
    PRINT "WAIT A MOMENT"
    60 LF=14
    OPENLF,UT,15,"I" + STR$(DT)
    OPEN5,UT,5,"I"
    GOSUB300
    FORX=0 TO 60
      PRINTLF,"B-P "S,B
      GOSUB500
      75 GOSUB500
      PRINT " "
      PRINT "TRACK "T" SECTOR "S" T"
      80 FORL=0 TO 255
        PRINTS.CHRS(AX(L,X))
        NEXTL
        PRINTLF,"L2 "S,DT,T,S
      90 NEXTX
      GOSUB300
      CLDSELF
      CLDSE5
      NEXTC
      PRINT "COPY END"
      END
300 REM *****
305 REM **CHECK DISK ERROR**
310 REM *****
320 INPUTLF,EN,EM,ET,ES
IFEN=0 THEN RETURN
330 PRINT "DISK ERROR"
PRINTEN,EM,ET,ES
END
500 REM *****
505 REM ** ADD. T & S **
510 REM *****
515 S=S+1
IFT>8 AND T<18 AND S>20 THEN S=0
  T=T+1
  RETURN
510 IFT>17 AND T<25 AND S>18 THEN S=0
  T=T+1
  RETURN
520 IFT>24 AND T<31 AND S>17 THEN S=0
  T=T+1
  RETURN
530 IFT>30 AND T<36 AND S>16 THEN S=0
  T=T+1
  RETURN
540 RETURN
550 REM *****
560 REM ** WHICH DRIVE ? **
565 REM *****
570 INPUT "DRIVE 8100",D
IFT<20 THEN EN=0
580 D=CHRS(D+40)
RETURN
1000 REM *****
1005 REM **FORMAT NEW DISK**
1010 REM *****
1020 INPUT "DISK NAME",D$
INPUT "DISK 10",I$
1010 PRINT "INSERT BACKUP DISK"
PRINT "AND PRESS A KEY"
WAIT197,64,255
GETA$
1020 PRINT "WAIT A MOMENT"
1030 OPEN5,UF,15,"N" + STR$(DT) + " " + D$ + " " + I$
CLOSE5
RETURN

```


Utility e linguaggio macchina

DISASSEMBLATO DELLE ROUTINES DI LETTURA DELLE PORTE GIOCHI

CF5F	A7	Sono i dati che verranno usati per raggiungere le routine di lettura di Joystick, Paddle e Light Pen
CF60	A2	
CF61	B6	
CF62	B8	
CF63	DB	
CF64	DE	
CF65	CD	
CF66	C6	
CF67	LDA DC	Vettore in pagina zero (\$59) per le locazioni \$DC2A-\$DC2B riguardanti il Complex Interface Adapter CIA1
CF68	STA \$80	
CF69	STA \$69	
CF6F	LDA \$24	Vettore in pagina zero (\$6B) per le locazioni \$D419-\$D41A riguardanti il Sound Interface Device
CF71	STA \$6C	
CF73	LDA \$19	
CF75	STA \$B	
CF77	LDA \$D2	Vettore in pagina zero (\$6D) per le locazioni \$D13-\$D14 riguardanti il Video Interface Controller VIC2
CF79	STA \$E	
CF7B	LDA \$13	
CF7D	STA \$D	
CF7F	LDA \$CF	
CF81	STA \$C	Vettore per i JMP indirizzati nella locazione \$FC (parte alta dell'indirizzo)
CF83	STA \$E	
CF85	LDA \$E	Vettore di partenza dati per i JMP indirizzati \$CFE nelle locazioni \$FD e \$FE (parte alta e parte bassa)
CF87	STA \$D	
CF89	JSR \$B1AA	Il PPA1 è convertito in intero e messo in \$64 e \$65
CF8E	CPV \$65	Confronta V con il contenuto di \$65
CF90	BEQ \$CF9A	Se il cont. di \$65 è V allora CF9A
CF92	CPV \$60	V ha raggiunto il B?
CF94	BEQ \$CF91	Se sì allora esce
CF96	INY	Continua il conteggio
CF97	JMP \$CF8E	E ritorna da capo
CF9A	LDA (\$D),V	Carica in A l'indirizzo dato a partire da \$CF8E
CF9C	STA \$B	Completa il vettore per il JMP indirizzato
CF9E	JMP (\$D),V	Salta alla locazione così formata
CFA1	RTS	Se il numero introdotto dall'USR > 8 allora ritorna
CFA2	LDV \$80	Azzera il reg. V
CFA4	JMP \$CF99	Salta a \$CF99
CFA7	LDV \$21	Carica il reg. V con 1
CFA9	LDA (\$D),V	Legge e mette in A il JCV il cui numero è in V
CFAB	EOB \$FF	Simula una NOT logico con un OR esclusivo con 255
CFAD	RND \$1F	Macchina 1 bit 5,6,7,8 del reg. A
CFAF	TAY	Trasferisce A in Y
CFB2	LDA \$80	Azzera A
CFB2	JSR \$B391	Trasforma A,V in numero intero e lo mette nel PPA1
CFB5	RTS	Ritorna al BASIC
CFB6	LDV \$80	Azzera il reg. V
CFB8	JMP \$CFBD	Salta a \$CFBD
CFB9	LDV \$21	Carica V con 1
CFBD	LDA (\$D),V	Legge la Light Pen, la coordinata (X,Y) disegna da V
CFBF	TAY	Come dalla linea CFAB alla linea CFB2
CFCA	LDA \$80	
CFCC	JSR \$B391	
CFCD	RTS	Ritorna al BASIC
CFCE	LDV \$80	Definisce quale Paddle e da quale Port.
CFCA	JMP \$CFDF	Salta a \$CFDF
CFCD	LDV \$21	Definisce quale Paddle e da quale Port.
CFCE	LDX \$82	
CFD1	JMP \$CFDF	Salta a \$CFDF
CFD4	LDV \$80	Definisce quale Paddle e da quale Port.
CFD6	LDX \$82	
CFD8	JMP \$CFDF	Salta a \$CFDF
CFDA	LDV \$21	Definisce quale Paddle e da quale Port.
CFDD	LDX \$80	
CFDF	SEI	Disabilita l'IRQ
CFE2	LDA \$DC22	Il Data Direction Register è depositato in \$22
CFE3	STA \$22	
CFE5	LDA \$C2	Predefinisce il DDR per l'input
CFE7	STA \$DC22	
CFEA	STX \$DC22	Sceglie il tipo di Paddle definito da X
CFEE	NOP	
CFEF	NOP	
CF92	NOP	
CF91	LDA (\$B),V	Legge la Paddle definita da V
CF93	TAY	Come dalla linea CFAB alla linea CFB2
CF96	LDA \$80	
CF9A	JSR \$B391	
CF9B	LDA \$22	Riconferma il DDR
CF99	STA \$DC22	
CF9E	CLI	Risabilita l'IRQ
CF9F	RTS	Ritorna al BASIC

Per realizzare il programma si hanno quindi due possibilità. Se si possiede un monitor che accetti la notazione standard MOS 6502 si può digitare il secondo listato, altrimenti si dovrà utilizzare il primo. Il programma in Basic si compone di tre parti. La parte che carica in memoria le routine in L.M. (da 5 a 40), le linee di data che le rappresentano (da 5000 a 5200) e un esempio di utilizzo del modulo in L.M. (da 100 a 220). Quest'ultima, che in ogni modo non è

indispensabile, può essere omessa, costituisce una maschera continuamente aggiornata contenente i valori del Joystick, delle Paddles e delle coordinate X ed Y della Light Pen. Vediamo ora come, una volta caricato il primo o il secondo programma, si utilizza il tutto. L'uso di queste routine particolarmente veloci è estremamente semplice. Basterà infatti definire una variabile uguale alla funzione USR (x). In essa la x sarà un numero da 1 a 8 che corrispon-

derà ai controllori come da tabella.

La USR potrà anche essere posta a seguito di un'altra istruzione Basic in quanto essa stessa rappresenta il valore del controllore desiderato che in pratica viene ad essa sostituito.

Poiché per l'uso dello statment USR il computer ha bisogno di conoscere l'indirizzo di partenza della routine in L.M. esso si deve memorizzare in due bytes adiacenti (785 e 786) nella notazione Low end High Address; è quello che fanno le linee 30 e 40.

Nel disassemblato, comunque commentato, sono ancora da notare due particolarità.

L'utilizzo di una loop di riconoscimento del numero passato al L.M. attraverso la USR che costruisce un JMP indiretto alle routine necessarie. L'utilizzo della subroutine di sistema che trasforma il numero contenuto nell'Accumulatore 1 in virgola mobile in un numero intero e che lo memorizza in \$64 e \$65. La routine è richiamata in \$B1AA e JSR \$B1AA. La simulazione del NOT logico con un OR esclusivo con \$FF alla linea \$CFAB. Aggiungo infine che i programmi Basic sono stati da me listati con un programma che evidenzia i loop e che divide in linee successive gli statment facenti parte di una stessa linea Basic. Si devono perciò battere tutti insieme e separati dal due punti «:».

Una «vita» per giocare

Vi proponiamo un'altra elaborazione del gioco «Vita», realizzata da Antonio Morra di Milano per i Computer Commodore. Prima di presentare il programma apriamo una piccola parentesi per spiegare il meccanismo del gioco stesso. Inventato dal Matematico dell'università di Cam-

bridge, John Horton, questo gioco, che ha come unico giocatore il computer, è stato riproposto da moltissimi altri matematici. Si gioca con una scacchiera illimitata, formata da innumerevoli caselle, dentro le quali nascono e muoiono un numero infinito di automi unicellulari. Data una certa configurazione iniziale, le regole con cui gli automi nascono, muoiono e si riproducono sono le seguenti: ogni automa ha nove caselle circostanti, se almeno due delle otto non contengono un automa il primo muore, mentre muore anche se i vicini sono più di tre. Ogni automa con vicino altri tre genera un nuovo essere nel turno successivo. Le morti e le nascite avvengono contemporaneamente.

La versione presentata dal nostro lettore è scritta completamente in Basic e può quindi girare tranquillamente con qualsiasi computer Commodore (ed anche di altre marche se si traducono le poche istruzioni differenti).

Il gioco si svolge in una griglia di 28 per 18 caselle. I frequenti REM aiutano a capire il programma.

Ecco la spiegazione delle principali routines:

1300-1380: Cerca gli indirizzi di partenza e di arrivo della parte piena della griglia.

1150-1240: Indicizza le cellule vuote immediatamente vicine a quelle piene.

1000-1130: Routine per la stampa del tempo del gioco e dello «STOP» dopo ogni generazione. La continuazione avviene dopo aver premuto la barra di spazio oppure automaticamente dopo 10 secondi senza interventi.

10000-: Programma vero e proprio con formazione della griglia, input iniziali ed elaborazioni varie.

Il programma è stato costruito con un Commodore 64 ma, come già detto, è sufficiente eliminare i simboli del colore (righe 110, 220 e 10007) perché giri perfettamente su qualsiasi Commodore.

Un «Renumber» per il Vic 20

Abbiamo scelto questo programma di «Re-number», realizzato da **Giancarlo Ammirata**, perché pensiamo possa interessare un gran numero di lettori proprietari di un Commodore VIC 20.

Le linee di qualsiasi programma vengono memorizzate dal VIC a partire dalla prima locazione di memoria disponibile, sotto forma di numeri. Il primo numero è sempre uno 0 che ha lo scopo di comunicare al sistema operativo l'inizio di una nuova linea di programma. I successivi due numeri sono i byte di LINK. Moltiplicando il valore del secondo per 256 ed aggiungendo il valore del primo si ottiene l'indirizzo di memoria della successiva linea di programma. Il numero che si ottiene sempre moltiplicando il secondo byte per 256 ed aggiungendo il primo.

A questo punto iniziano le vere e proprie istruzioni del programma. Ogni lettera, numero, istruzione BASIC, segno di punteggiatura, ecc. viene tradotta dal VIC in un numero che corrisponde ad esso, risparmiando così una buona quantità di memoria. Vediamo, ad esempio, come viene immagazzinata in memoria la linea:

```
100  A=A+1:IFA>256
    THENA=0
```

Supponiamo di non avere espansioni di memoria. La prima locazione di memoria disponibile sarà allora la 4906:
4906:0
4907:22 4907:16 > Byte di LINK - $16 \times 256 + 12 = 4118$ (indirizzo d'inizio della successiva riga di progr.)
4909:100 4100:0 > Numero di linea
4101:65 > lettera A
4102:178 > segno =
4103:65 > lettera A
4104:170 > segno +

4105:49 > valore 1

4106:58 > segno :

eccetera.

Alterando quindi le opportune locazioni di memoria è possibile cambiare i valori dei vari numeri di linea. Il programma da renumerare deve necessariamente avere i numeri di linea seguenti le istruzioni del tipo GOTO, THEN, GOSUB, ecc. composti da cinque cifre. Così il numero 100 deve diventare 00100.

Il problema sorge dal fatto che se il nuovo numero è più lungo di quello già esistente bisognerebbe spostare verso il basso tutte le locazioni di memoria che seguono per far posto al (o ai) caratteri in più. Suggestivo, eventualmente, di usare una piccola routine in linguaggio macchina, poiché il basic è troppo lento.

Vediamo ora come opera in dettaglio il programma:

linee

60006-60050

Conta il numero di linee del programma da renumerare

60060-60090

Crea il vettore A () con i vecchi numeri di linea

60100-60280

Controlla i rimanenti numeri di linea seguenti le istruzioni tipo GOTO, GOSUB THEN

60290-60310

Incrementa del passo il nuovo valore del numero di linea. Il programma gira con il VIC 20 senza espansioni. In caso contrario basta variare il valore della variabile P alle linee 60000, 60060, 60100 con quello d'inizio della memoria libera con la relativa espansione (1024 con la 3K, 4608 con la 8 o la 16K).

Per variare il passo di renumerazione basta variare PAS alla linea 60100. Ecco l'elenco delle variabili usate:
P = inizio memoria libera
C = Numero di righe di cui è formato il programma da renumerare
BL = Byte di LINK
NL = Numero di linea considerato

Listato - Vita per giocare

```
10060 PRINT"TI"
10066 PRINT"
10070 FORW=1TOVOM
10080 PRINTW"" CELLULA "
10090 INPUT"RIGA" ,A$
10095 IFVAL(A$)<1ORVAL(A$)>18THENPRINT"J":GOTO10090
10100 PRINT"J"
10110 INPUT"COLONNA" ,B$
10115 IFVAL(B$)<1ORVAL(B$)>28THENPRINT"J":GOTO10110
10120 PRINT"J"
10130 PW=VAL(A$):CW=VAL(B$):IF VIZ(RW,CW)=1THENPRINT"■",GOTO10090
10140 VIZ(RW,CW)=1:GOSUB1150:GOSUB1300
10150 PRINT"■";PRINTSPC(CW):FORW=1TORW:PRINT" ";
10160 NEXT:PRINT" "
10170 PRINTSP$ NEXT
10180 PRINTSP$ " ***** 1° GENERAZIONE ***** " NG=1
10190 TIME$="000000":GOSUB1050
10200 :
10210 :
10220 :
10230 :
10240 :
10500 REM ***SCANNING RETICOLO*****
10510 FG$="":FORRW=1RZTOFRZ:FORCW=1CZTOFCZ:IFVIZ(RW,CW)=0THEN12000
10600 TVZ=0:IFPW=1THEN10640
10610 IFCW<15THENVI=VIZ(RW-1,CW+1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM AD <-----
10620 VI=VIZ(PW-1,CW):IFVI=1ORVI=2THENTVZ=TVZ+1:REM AC <-----
10630 IFCW>1THENVI=VIZ(RW-1,CW-1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM AS <-----
10640 IFCW>1THENVI=VIZ(RW,CW-1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM FS <-----
10650 IFCW<28THENVI=VIZ(RW,CW+1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM FD <-----
10655 IFTVZ>3THEN10700
10660 IFRW=18THEN10700
10670 IFCW>1THENVI=VIZ(RW+1,CW-1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM BS <-----
10680 VI=VIZ(RW+1,CW):IFVI=1ORVI=2THENTVZ=TVZ+1:REM BC <-----
10690 IFCW<28THENVI=VIZ(RW+1,CW+1):IFVI=1ORVI=2THENTVZ=TVZ+1:REM BD <-----
10700 REM
10710 VI=VIZ(RW,CW)
10720 REM SE MUORE
10730 IFVI=4THEN10770
10740 IFTVZ<2ORTVZ>3THENVIZ(RW,CW)=2:FG=1:GOSUB1300
10750 REM SE GENERA
10760 IFVI<4THEN12000
10770 IFTVZ=3THENVIZ(RW,CW)=3:FG=1:GOSUB1150:GOSUB1300
10780 :
10820 :
12000 NEXT:GOSUB1050:NEXT:GOSUB1050
12010 REM QUADRO MUTAZIONI
12020 FORRW=1RZTOFRZ:FORCW=1CZTOFCZ
12030 IFVIZ(RW,CW)=0THEN12060
12040 PRINT"■";PRINTSPC(CW):FORW=1TORW:PRINT" ";NEXT
12050 PRINTC$(VIZ(RW,CW))
12060 NEXT:GOSUB1050:NEXT:GOSUB1050:GOSUB1080
12080 :
12090 :
12100 REM QUADRO NUOVA GENERAZIONE
12110 FORRW=1RZTOFRZ:FORCW=1CZTOFCZ
12120 IFVIZ(RW,CW)=0THEN12160
12130 PRINT"■";PRINTSPC(CW):FORW=1TORW:PRINT" ";NEXT
12140 IFVIZ(RW,CW)=1ORVIZ(RW,CW)=3THENPRINTC$(1):VIZ(RW,CW)=1
12150 IFVIZ(RW,CW)=2THENPRINT" ":VIZ(RW,CW)=4
12160 NEXT:NEXT:GOSUB1050:NG=NG+1
12170 PRINTSP$ " ***** NG ***** "
12180 IFFGTHENFG=0:GOTO10500
12190 PRINTSP$ " FINE EVOLUZIONE " " :END
```

A () = Vettore con i vecchi numeri di linea
AL = Byte alto del numero di riga
BA = Byte basso del numero di riga
IN = Numero di riga di partenza
PAS = Passo d'incremento

CO = lunghezza del numero da cambiare
VS = Stringa contenente il valore da cambiare
V = Valore da cambiare
MN = Posto nel vett. A () dove si trova il valore da cambiare
NV = Valore nuovo da so-

stituire
Questo programma va digitato in coda al programma da renumerare, il quale deve avere i numeri di riga più bassi di 60000.

Per farlo girare basta dare un RUN60000.


```

60000 REM*****
60001 REM# REMNUMBER #
60002 RFM# B# #
60003 REM#GIANCARLO AMMIRATA#
60004 REM# PER F. C. CLUS #
60005 REM*****
60006 F=4096 C=-1
60010 DEFFNBL(X)=PEEK(X+2)*256+PEEK(X+1)
60020 DEFFNBL(X)=PEEK(X+4)*256+PEEK(X+3)
60030 BL=FN BL(P)-1 NL=FNBL(F)
60040 IFNL>60000THEN60060
60050 F=BL C=C+1 GOTO60030
60060 DIMA(C-1) F=4096
60070 FORI=0TOC-1
60080 BL=FNBL(F)-1 NL=FNBL(F)
60090 A(I)=NL F=BL NEXTI
60100 F=4096 AL=00 BA=10 IN=BA-FA5=10
60110 FORT=1TOBL BL=FNBL(P)-1
60120 POKEF+4,AL POKEP-3,BA
60130 FORJ=PTOBL CQ=0 V$=""
60140 IFPEEK(J)<137ANDPEEK(J)<141ANDPEEK(J)<167THEN60280
60150 IFPEEK(J+1)<40ORPEEK(J+1)>57THEN60280
60160 FORK=1TO5
60170 V$=V$+STR$(PEEK(J+1)-48) J=J+1 V=VAL(V$) CQ=CQ+1
60180 IFPEEK(J+1)<40ORPEEK(J+1)>57THENMN=1 GOTO60200
60190 NEXTK
60200 FORK=0TOC-1
60210 IFACK=JTHEN60230
60220 MN=MN+1 NEXTV
60230 NV=IN+FA5*(MN-1) NV$=STR$(NV)
60240 FORL=2TOLEN(V$)
60250 POKEJ-CQ+L-1,VAL(MID$(NV$,L,1))+48
60260 NEXTL
60270 IFLEN(NV$)=57THEN60280
60280 FORL=LEN(NV$)+1 POKEJ-CQ+L,32 NEXT
60290 NEXTJ
60300 F=BL BA=BA-FA5
60310 IFBA>255THENBA=BA-256 AL=AL+1 GOTO60300
60320 NEXTI
READY.

```

```

10 REM ALLUNAGGIO
20 REM BY LUNA A " T " E P " FC CLUB
30 P " M ISTRUZIONI
40 PRINT"CLR/HOME"
50 PRINT"DEVI UNIRTI A L'ALLUNAGGIO IN UNA NAVICELLA"
60 PRINT"CH- SI TROVA A 120 KM D'ALTEZZA SULLA LUNA"
70 PRINT"CON VELOCITA' INIZIALE DI 3600 KM/H; HAI A "
80 PRINT"DISPOSIZIONE, OGNI 10 SECONDI, DA 8 A 200 "
90 PRINT"EG DI CARBURANTE PER LA FRENATA (PUOI AN-"
100 PRINT"CH- RINCHIARVI E PROSEGUIRE IN CAPOTA LI-"
110 PRINT"BERA); LA COTA INIZIALE E' DI 3600 KG, "
120 PRINT"BUONA FORTUNA! PREMIL TASTO PER CONTINUARE"
130 GET AS: IF AS = " " THEN GOTO 140
140 PRINT"CLR/HOME"
150 PRINTTAB(5)"SEC";TAB(10)"ALT";TAB(15)"VEL";TAB(20)"CARB";TAB(25)"SERB"
160 PRINT
170 REM INIZIALIZZAZIONE DEI CONTROLLI INIZI LI
180 CR=16000;T=0;V=0;S=120
190 INP"TKG";KG
200 IF (KG<0) AND ((KG<8.12)>200) THEN 190
210 AC=0.16*(16-(KG/5000))
220 CR=CR-KG
230 IF CPG THEN 300 : REM CONTROLLO LIVELLO SERBATOIO
240 FOR I=0 TO 9.9 STEP .1 : REM INIZIO DEL CICLO DI DIECI SECONDI
250 VE=V+AC*I : REM CALCOLO VELOCITA'
260 AL=120-(VE*I+AC*(I*I)/2)/500 : REM CALCOLO ALTEZZA
270 IF AL<0 THEN 350 : REM CONTROLLO ALTIMETRO
280 PRINT"(CURSOR UP)"
290 PRINTTAB(5)INT((I*10)/10);TAB(10)INT((AL*100)/100);TAB(15)INT((VE*100)/360);TAB(20)INT(KG);TAB(25)CR
300 NEXT I
310 T=T+10;S0=AL;V0=VE : REM AGGIORNAMENTO VARIABILI
320 GOTO 190
330 PRINT"HAI FINITO IL CARBURANTE A DISPOSIZIONE !"
340 PRINT"SEIAA";AL;"KM DI ALTEZZA... ADDIO... ":GOTO 400
350 IF VE 10 THEN 380
360 PRINT"SRI ALLUNATO DOPO";T+I;"SECONDI ALLA VELOCITA'"
370 PRINT"DI";VE;"KM/H LA NAVICELLA E' DISTRUTTA!";GOTO 400
380 PRINT"SEI FELICEMENTE ALLUNATO DOPO";T+I;"SECONDI"
390 PRINT"HAI CONSERVATO";CR;"KG DI CARBURANTE; COMPLIMENTI!"
400 PRINT"VOI CONTINUARE?(S/N)"
410 GET AS: IF AS = " " THEN 410
420 IF AS = "E" THEN 140
430 STOP

```

Speciale Commodore 43

Leggere i diversi floppy

E vi sono dei problemi d'incompatibilità fra questi differenti supporti. In certe applicazioni può tornar utile disporre di una procedura che riconosca ciascun tipo di dischetto evitando quindi possibili errori. Come proce-

```

LISTATO, ****
10 REM DEFINIZIONE DELLA UNITA' DISCHETTI
20 REM
100 OPEN#0,8,15,"I0"
120 OPEN#0,8,7,"00"
130 GET#0,0,CLOSE#0
140 IF A$=CHR$(1) THEN PRINT "CBM 3040":END
150 IF A$="A" THEN PRINT "CBM 4040":END
160 PRINT "CBM 8050":END

```

HOME DEL DISCHETTO	CBM 3040 4040	CBM 8050
10	143 - 156	5 - 20
	161-162	223-24

```

10 REM LETTURA DEL NOME O DELLA - ID - DI UN DISCHETTO
100 N=5:REM PER CBM 8050 ALTIMENTI N=143
110 OPEN80,8,15,"I0"
120 OPEN86,8,6,"#0"
130 FOR I=1 TO N-1:GET#86,A$:NEXT I
140 N$="" FOR I=1 TO 16
150 GET#86,A$:IF A$=CHR$(160) THEN 170
160 N$=N$+A$
170 NEXT I
180 GET#86,A$:A$
190 I$="" FOR I=1 TO 2
200 GET#86,A$:IF A$=CHR$(160) THEN 220
210 I$=I$+A$
220 NEXT I
230 PRINT "NOME = " N$." ID = ",I$
240 END

```

[illegible]

44 Speciale Commodore

dere se un programma lavora su questa unità floppy o su quest'altra? Leggendo la **Directory (\$0 o \$1)**. Nel primo carattere letto si determinerà su quale unità è stato formattato il dischetto:

CBM 3040 CODE ASCII
PRIMO CARATTERE = 1
CBM 4040 CODE ASCII
PRIMO CARATTERE = 65(=«A»)

CBM 8050 CODE ASCII
PRIMO CARATTERE = 67(=«C»)

Il listato 1 vi dimostra quanto asserito.

Questo metodo non determina direttamente su quale unità di dischetto si sta lavorando, ma su quale supporto è stato formattato. Attenzione però, è pericoloso scrivere su un dischetto 4040 con un 3040. Il metodo che vi presentiamo è particolarmente utile in questo caso. Quando avviene la formattazione si fornisce al sistema un nome (fino a sedici caratteri) ed un identificatore *ID* di due caratteri.

Quando il nome è formato da meno di 16 caratteri esso viene completato da «Shift Space», code Ascii: 160.

La stessa cosa è valida per la - ID - per questo motivo si sconsiglia di mettere alcuni «Shift Space» nel nome o nella ID di un dischetto. Per leggere la ID faremo uso del programma con listato 2, nulla ci vieta di combinare il programma di listato uno con quello di listato 2.

In queste procedure viene utilizzato il Drive 0, per il Drive 1 bisogna sostituire I0 e \$0 con J1 e \$1.

A caccia di stelle sul VIC 20

Un bel programma di conquista stellare, abbastanza corto ma ricco di interesse, va segnalato senz'altro ai lettori anche se le istruzioni del listato sono in lingua inglese.

L'obiettivo del gioco è di andare a zonzo per un univer-

so popolato di stelle, demolendo tutto quello che capita sotto mano: ma naturalmente le cose non filano così lisce, perché due astronavi nemiche vi daranno la caccia e semineranno mine vaganti per cercare di fermarvi.

Ultimo dettaglio: la seconda astronave nemica entrerà in gioco solo dopo che avrete completato la distruzione delle prime otto stelle.

Le istruzioni di gioco e i comandi da usare sono scritti a video all'inizio (righe 2-5 del listato): buon divertimento e occhio alle mine! (L.F.)

Sorting col metodo «Bubble Sort»

L'ordinare dei punti numerici o alfabetici è un compito classico per un computer.

Chiaramente l'ordinamento può essere fatto in ordine crescente od in ordine decrescente.

Per ordinare dei dati esistono molti metodi; in questo numero di Personal Computer vedremo quali sono e quali usare secondo la capacità del computer a nostra disposizione.

Anzitutto ordinare dei dati vuol dire cambiar loro di posizione fisica o logica e per far questo usiamo la tecnica dello scambio. Il procedimento per tale scambio è il seguente:

Confronto Dato (I) con Dato (L) se sono già ordinati proseguo con incrementare I, altrimenti scambio, cioè:

SAVE = DATO (I)
DATO (I) = DATO (L)
DATO (L) = SAVE

Uso cioè una variabile temporanea.

L'ordinamento Bubble Sort è il più diffuso ma non è né il più veloce, né quello che occupa meno memorie, ma è il più facile da capire.

Listato - Bubble Sort

```
10 REM BUBBLE SORT
20 INPUT "Numero di dati ";N
30 DIM DATO(N)
40 FOR I=1 TO N
50 INPUT "DATO ";DATO(I)
60 NEXT I
70 REM ORDINAMENTO
80 FOR I=1 TO N-1
90 FOR L=I+1 TO N
100 IF DATO(I)>=DATO(L) THEN SWP
110 REM SCAMBIO
120 SAVE=DATO(I)
130 DATO(I)=DATO(L)
140 DATO(L)=SAVE
150 NEXT L
160 NEXT I
```

Problemi di listato

Cercheremo, in questo articolo, un nuovo modo di listare i programmi per i computers Commodore. Il problema deriva dal fatto che la Commodore usa caratteri speciali, non ASCII, per rappresentare importanti comandi, colori, caratteri e così via. Non è sempre possibile in un listato stampato per differenziare tra i caratteri grafici speciali, uguagliare con stampanti specificamente costruite per questi computer. Per esempio; una linea verticale sottile in un solido blocco nero, può rappresentare alcune funzioni differenti su un Commodore. Piuttosto che cercare di rappresentare la grafica speciale di questi computers, abbiamo stampato un testo che descrive la funzione. Per esempio la funzione HOME

sui Commodore 64/VIC 20, che sposta il cursore nell'angolo alto a sinistra dello schermo, è rappresentata graficamente con una S in campo inverso. Proprio quando questa S in campo inverso è «leggibile» dal listato, non rappresenta quello che è accaduto. I nostri listati rimpiazzano la singola S in campo inverso con la stringa (HOME). Il cuore in campo inverso, che rappresenta la funzione pulisci video, può essere listato come (CLEAR).

Una lista completa di queste conversioni di listaggio è riportata di seguito all'articolo per rendere facile le traslazioni. Il listato delle linee che hanno caratteri grafici speciali nel generatore di caratteri EPSON FX 80, sono presentati sotto nel nuovo formato.

Spero che questa nuova tecnica possa rendere i programmi più comprensibili e facili da usare.

Note - Problemi di listato

- 1) rappresenta il tasto SHIFT.
- 2) = rappresenta il tasto Commodore nella parte bassa a sinistra della tastiera
- 3) CTRL rappresenta il tasto CTRL
- 4) i caratteri grafici sono rappresentati nel listato dai tasti battuti, richiesti per generare il carattere
- 5) un numero direttamente dopo un (SIMBOLO), indica i multipli del SIMBOLO: (DOWN6) significa battere DOWN per 6 volte.

Problemi di listato

Tabella dei comandi

C/674 - Tastiera

FUNZIONI - Tasti funzione programmabili

(CLEAR)	~ CLR	;pulitura video
(HOME)	HOME	;ritorno cursore a casa
(INSERT)	~ INST	;inserimento carattere
(DOWN)	CRSR DOWN	;cursore giù
(UP)	~ CRSR UP	;cursore su
(RIGHT)	CRSR RIGHT	;cursore a destra
(LEFT)	~ CRSR LEFT	;cursore a sinistra

(F1)	f1
(F2)	~ f2
(F3)	f3
(F4)	~ f4
(F5)	f5
(F6)	~ f6
(F7)	f7
(F8)	~ f8

COLORI

(BLACK)	CTRL 1 BLK	;nero
(WHITE)	CTRL 2 WHT	;bianco
(RED)	CTRL 3 RED	;rosso
(CYN)	CTRL 4 CYN	;blu verde
(PURPLE)	CTRL 5 PUR	;porpora
(GREEN)	CTRL 6 GRN	;verde
(BLUE)	CTRL 7 BLU	;blu
(YELLOW)	CTRL 8 YEL	;giallo
(RVS)	CTRL 9 RVS ON	;invereo colori
(RVS OFF)	CTRL 0 RVS OFF	;fine invereo colori
(ORANGE)	= 1	;arancio
(BROWN)	= 2	;marrone
(GREY 1)	= 3	;grigio 1
(GREY 1)	= 4	;grigio 1
(GREY 2)	= 5	;grigio 2
(LT GREEN)	= 6	;verde chiaro
(LT BLUE)	= 7	;azzurro
(GREY 3)	= 8	;grigio 3

CHARACTERI SPECIALI

(PI)	n ~ Pi Greco
(POUND)	£ Sterlina
(UP ARROW)	↑ Freccia su
(BACK ARROW)	← Freccia indietro

COMANDI NON PRESENTI SULLA TASTIERA

(DIS=)	CHRS (8)
(END=)	CHRS (9)
(LOWER CASE)	CHRS (14)
(UPPER CASE)	CHRS (142)
(RETURN)	CHRS (142)
(DEL)	CHRS (20)
(SPACE)	CHRS (160)
NOTE	

Crittografare i programmi

Questo programma, realizzato da Massimo Roberto Garibaldi, permette la codifica di altri programmi scritti in Basic.

Può essere utile quando si deve lasciare copie di programmi riservati alla portata di estranei.

Il programma gira su un VIC 20, ma può essere facilmente trasportato su qualunque Personal che ne utilizzi il sistema di memorizzazione delle linee Basic.

Tale metodo è basato sulla trasformazione delle singole istruzioni in numeri, chiamati TOKEN, in base ad una tabella memorizzata nel computer e riportata sul manuale dello stesso.

Il programma da me ideato esegue un OR esclusivo di tutti i token del programma, in base ad una chiave alfanumerica che l'operatore dovrà inserire sia durante la codifica che la decodifica.

Il programma inoltre è completamente trasparente all'utente, in quanto si adatta automaticamente alla configurazione di memoria di cui il VIC può essere dotato. Questo è reso possibile dalle linee 1 e 2 del listato, che calcolano rispettivamente la fine del programma e la fine della memoria RAM disponibile.

I valori ottenuti in base ai puntatori del VIC, vengono poi usati alla linea 3 per spostare la fine della memoria e proteggere quindi alcuni valori indispensabili all'esecuzione corretta del programma seguente. Il cuore del programma sta tra la linea 200 e 220. In particolare è la 210 che si occupa della codifica e decodifica vera e propria.

Come già detto essa consiste in un OR esclusivo tra i tokens del programma ed ogni carattere della chiave di codifica. L'OR esclusivo, che non esiste nel Basic del VIC, è stato reso mediante una

differenza tra l'OR e l'AND dei valori da manipolare.

Sempre in linea 210, con l'ultima istruzione, viene esclusa la possibilità che tali valori siano eguali, nel qual caso si avrebbe un'errore nella codifica.

È da aggiungere il fatto che il programma assiste l'utilizzatore dall'inizio alla fine. È infatti completamente automatico l'accodamento del programma da modificare al programma principale che, terminata la codifica, si autodistrugge, permettendo di salvare direttamente su nastro o disco il programma cifrato.

L'orologio sveglia con il C64

Prendo spunto dalla lettera di Massimo Roberto Garibaldi per pregare cortesemente i nostri lettori di allegare al listato del programma una adeguata spiegazione del suo funziona-

mento, sia per aiutare i lettori a capirne il meccanismo (copiare pedestramente un programma non aiuta sicuramente ad imparare a programmare) sia per aiutare il sottoscritto a verificarne il funzionamento.

Passiamo ora al programma del nostro lettore.

Il listato tratta di un orologio sveglia che si inserisce in permanenza nell'angolo superiore destro dello schermo e può quindi essere utilizzato durante l'esecuzione o la realizzazione di programmi per tener conto del tempo impiegato, oppure come sveglia vera e propria. Il programma, interamente in linguaggio macchina, basa il suo funzionamento sulla modifica delle routine di interrupt permettendo il continuo aggiornamento basato sul clock esterno.

A questo punto aggiungiamo due righe: può accadere che, anche battendo correttamente il programma, quando si imposta SYS 832,

Problemi di listato

```

10 PRINT "(CLEAR) GENERATORE DI CARATTERI"
1010 BY=FX+13*CH:PRINT "(HOME)";
1020 FOR J=7 TO 0 STEP -1:K=2(UP ARROW)J
1440 FOR J=7 TO 0 STEP -1:K=2(UP ARROW)J
2015 PRINT "(HOME,RIGHT8)CARATTERI #";CH;"(LEFT) (SPACE,HOME)";
2040 IF TS="(POUND)" THEN CP=225:GOTO 2025
2050 IF TS="(UP ARROW)" THEN CP=87:GOTO 2025
3000 PRINT "(HOME)";QQ$;LL$;"(RIGHT)";INPUT CH
3020 PRINT "(HOME)";QQ$;LL$;S9$
4000 PRINT "(CLEAR)";QQ$;" + ACCESO / - SPENTO / (POUND) MEZZO /
      (UP ARROW) PIENO"
4010 PRINT"(DOWN)F1 LEGGE CARATTERE #"
5130 PD=ASC("."):PRINT"(CLEAR,BLACK)";
5160 CH=0:F1$="(F1)":F2$="(F2)":F3$="(F3)"
5165 F4$="(F4)":F5$="(F5)":F6$="(F6)":F7$="(F7)"
5170 CL$="(RIGHT)":CR$="(LEFT)":CU$="(UP)"
5175 CD$="(DOWN)":HM$="(HOME)":CS$="(CLEAR)"
5180 ZL$=RT$+"(UP,RIGHT14)"
5190 ZR$="(LEFT17)"
5196 QQ$="(DOWN10)"
5197 LL$="(RIGHT18)"
6010 PRINT"(HOME,DOWN17)";
6210 PRINT"(HOME,DOWN16)";
6300 PRINT"(UP)";EL$;B$;" CARICATO";:GOTO 2010
6600 PRINT RT$;"(UP,RIGHT19)";
6604 PRINT"(LEFT18)";
7200 PRINT QQ$;"(DOWN)";
7210 PRINT RT$;"(RIGHT16)";
7220 PRINT CH;"(LEFT) (LEFT7)";

```


Listato - Crittografare i programmi

CRITTOGRAFIA
BY M.R.G. SOFTWARE
(c) 1983

```
0 PRINT"ON CRITTOGRAFIA "
1 I=PEEK(45)+PEEK(46)*256-2:H=INT(I/256):L=I-H*256
2 F=PEEK(55)+PEEK(56)*256-4:HF=INT(F/256):LF=F-HF*256
3 POKE51,LF:POKE55,LF:POKE52,HF:POKE56,HF:POKEF,PEEK(43):POKEF+1,PEEK(44):POKEF+
2,L:POKEF+3,H
4 PRINT"METTI LA CASSETTA NEL REGISTRATORE POI PREMIO TASTO":WAIT197,255,64
5 PRINT"PG43,"L":PG44,"H":LF":L=PEEK(F):H=PEEK(F+1):PRINT"PG43,"L":PG44,"H"
:GF20"
10 POKE631,19:POKE632,17:POKE633,17:POKE634,31:POKE635,13:POKE636,19:POKE637,13:
POKE198,7:END
20 INPUT"CHIAVE";K$
25 IFK$=""THEN20
30 F=PEEK(55)+PEEK(56)*256
40 K=PEEK(F+2)+PEEK(F+3)*256+4:C=1
50 GOSUB200:K=K+1:C=C+1
60 IFPEEK(K)=0THENK=K+5:IFPEEK(K-3)=0ANDPEEK(K-4)=0THEN100
80 GOT050
100 F=PEEK(55)+PEEK(56)*256:L=PEEK(F+2):H=PEEK(F+3):POKE43,L:POKE44,H:END
200 IFLEN(K$)<CTHENC=1
210 P=PEEK(K):P=(PORASC(MID$(K$,C,1)))-(PANDASC(MID$(K$,C,1))):IFP=0THENRETURN
220 POKEK,P:RETURN
```

READY.

non si veda comparire alcun orologio. Niente paura, non è il programma che non funziona, sono solo le cifre dell'orologio che hanno lo stesso colore dello sfondo del teleschermo, e quindi è sufficiente digitare POKE 982, n (n = numero del colore desiderato).

Il gioco «Isola» per VIC20 e C64

«Isola» è un grande gioco di strategia statica proposto su Personal Computer n. 1/1983, ed è commercializzato in Italia dalla Ravensburger. L'obiettivo del gioco è di

«isolare» l'avversario muovendo sulla apposita scacchiera come un re di scacchi. Le dimensioni della tavola da gioco sono a scelta dei giocatori e variano da un minimo di 7 x 5 a un massimo di 9 x 7 caselle.

La particolarità cui «Isola» deve il suo nome sta nel fatto che una volta effettuata la mossa, la casella di provenienza viene cancellata dal gioco e resa intransitabile. Il programma richiede i nomi dei giocatori, identificati da simboli diversi, e gestisce le mosse, controllando che le richieste siano lecite: i commenti nei programmi e nelle mappe video spiegano il resto, tuttavia è il caso di ricor-

dare che è possibile muovere solo su caselle ancora transitabili, non è lecito calpestare l'avversario e infine perde chi resta senza mosse valide.

Al termine della partita il calcolatore si complimenta con il vincitore.

Vista anche la dimensione e la semplicità dei listati, nella versione del nostro amico di vecchia data Andrea Acquaviva di Rovigo, vi suggeriamo di provare questo gioco, che ha riscosso molti consensi ed è stato considerato uno dei migliori proposti lo scorso anno. Buon divertimento!

(L.F.)

Personal
Computer
ti aspetta
in edicola
l'1/1/85
con
succose
novità.

Listato Orologio sveglia

```

110 GOSUB210
120 PRINT"*** IL TEMPO NON ASPETTA ***"
130 PRINT"ISTRUZIONI PER L'OROLOGIO:"
140 PRINT"SYS 832      :ACCENDE
150 PRINT"SYS 994      :SPEGNE
160 PRINT"POKE 982,N   :CAMBIA COLORE

180 PRINT"F1          :SPEGNE SVEGLIA"
190 GOSUB360:END
200 REM LE ROUTINES POSSONO ESSERE USATE INDIPENDENTI
210 CH=0:FORI=832TO1008
220 READA:POKEI,A:CH=CH+A:NEXT
230 FORI=679TO742:READA:POKEI,A:CH=CH+A:NEXT
240 IFCH=23588THENPRINT"QWHA...DATA ERROR":STOP:NOTE CHECKSUM
250 INPUT"***** AM O PM ?";A$:INPUT"ORA ?";H
260 PRINT"*** DIGITA IL MINUTO DI PARTENZA"
270 PRINT"  PREMI RETURN PER INIZIARE"
280 IFH>12THENA$="P":H=H-12:GOTO280
290 IFH>9THENH=H+6:REM CONVERSIONE IN BCD
300 IFLEFT$(A$,1)="P"THENH=H+128
310 C=56328:POKEC+3,H:POKEC+1,0
320 INPUTM:M=M+INT(M/10)*6
330 POKEC+2,M:POKEC,0:SYS832:PRINT"*** SE NON VA BENE, PREMI UN TASTO
340 FORI=1TO1000:IFPEEK(198)THENPOKE198,0:SYS994:GOTO250
350 NEXT:RETURN
360 PRINT"QUANDO VUOI LA SVEGLIA?"
370 INPUT"AM O PM";A$:A$=LEFT$(A$,1)
380 INPUT"A CHE ORA?";H
390 IFH>12THENA$="P":H=H-12:GOTO390
400 H=H-6*(H>9)-12*(A$="P"):REM CONVERTE IN BCD E METTE INDICATORE AM/PM
410 INPUT"A CHE MINUTO";M
420 M=M+INT(M/10)*6
430 C=56328:POKEC+7,136:POKEC+3,H:POKEC+2,M:POKEC,1:POKEC+7,8:REMSVEGLIA
440 POKE54273,99:POKE54278,240:POKE54276,21
450 POKE54287,2:POKE54290,17:REM SUONO
460 RETURN
470 DATA120,173,20,3,162,89,234,234,234,142,20,3,173,21,3,162,3
480 DATA234,234,234,142,21,3,88,96,173,11,220,170,41,15,24,105,48
490 DATA141,67,4,138,16,4,162,16,16,2,162,1,142,77,4,162,32
500 DATA41,16,240,2,162,49,142,66,4,173,10,220,170,41,15,105,48
510 DATA141,70,4,138,74,74,74,74,24,105,48,141,69,4,173,9,220
520 DATA170,41,15,105,48,141,73,4,138,74,74,74,74,24,105,48,141
530 DATA72,4,173,8,220,105,48,141,75,4,169,32,141,65,4,141,76
540 DATA4,141,79,4,162,14,157,24,4,202,208,250,169,58,141,68,4
550 DATA141,71,4,169,46,141,74,4,169,13,141,78,4,169,1,162,13
560 DATA157,65,216,202,208,250,76,167,2,120,169,49,234,141,20,3,169
570 DATA234,234,141,21,3,88,96
580 DATA173,13,220,41,4,240,3,141,225,2,173,225,2,240,40,165,162
590 DATA106,106,106,41,12,141,32,208,41,4,141,24,212,240,11,162,5
600 DATA189,225,2,157,33,4,202,208,247,165,197,201,4,208,6,142,225
610 DATA2,142,24,212,76,49,234,0,1,12,1,18,13

```

READY.

Listato «Isola» per VIC 20

```

10 REM *****
20 REM *
30 REM *      I S O L A      *
35 REM *
40 REM *  PERSONAL COMPUTER CLUB  *
45 REM *
50 REM *      VERSIONE VIC 20      *
70 REM *  DI ANDREA ACQUARVIVA    *
80 REM *
90 REM *****
100 REM
110 REM * VARIABILI STRINGA PER I SIMBOLI DELLE DIREZIONI *
120 M$(1)="↑" : M$(2)="↖" : M$(3)="↗" : M$(4)="←"
130 M$(5)="→" : M$(6)="↘" : M$(7)="↙" : M$(8)="↕"
140 C$=" "
150 PRINT "I S O L A BY P.C CLUB"
155 REM * ISTRUZIONI INIZIALI *
160 PRINT "PER MUOVERE USARE : "
170 PRINT "1 = "M$(1)" 2 = "M$(2) : PRINT "3 = "M$(3)
180 PRINT "4 = "M$(4) : PRINT "5 = "M$(5) : PRINT "6 = "M$(6)
190 PRINT "7 = "M$(7) : PRINT "8 = "M$(8) : PRINT
195 PRINT "PREMI UN TASTO"
200 GETR$: IFR$="" THEN 200
210 REM * RICHIESTA NOMI E DIMENS. SCACCHIERA *
220 PRINT "I S O L A BY P.C CLUB"
230 PRINT "NOME DEI GIOCATORI" : PRINT
240 PRINT : INPUT "1° " : P1$
250 PRINT : INPUT "2° " : P2$
270 PRINT "SCEGLI LE DIMENSIONI DELLA SCACCHIERA " : PRINT
280 PRINT " ( MIN.7X5, MAX.9X7 ) " : PRINT
290 INPUT "ORIZZ. " : X
300 PRINT : INPUT "VERT. " : Y
990 REM * DISEGNO DELLA SCACCHIERA E DEI SIMBOLI *
1000 PRINT "I S O L A BY P.C CLUB"
1200 A=7746 : B=(X*2-2) : C=38468
1250 FORR=1 TO Y : FORI=0 TO B STEP 2 : POKER+1 : 102 : POKED+1 : 5
1350 NEXT
1400 A=A+44 : C=C+44
1450 PRINT : PRINT
1500 NEXT
1550 G1=7680 : G2=7634+I
1600 POKE G1,90 : POKE G2,63
1640 GOSUB 3500 : PRINT "CHI GIOCA PER PRIMO " : PRINT
1650 INPUT "1/0/2 " : P$
1660 PRINT " " : C$
1670 P=VAL(P$)
1680 IF P<1 OR P>2 THEN 1640
1690 IF P=2 THEN 1710
1695 REM PRINCIPALE *** 4 LINEE PER LA GESTIONE DEL GIOCO ***
1700 T=G1 : GOSUB 3500 : FORR=1 TO Y : PRINT C$ : NEXT : GOSUB 3500 : PRINT "MUOVE (♦)" : P1$
1705 PRINT : PRINT "MOSSE " : GOSUB 2500 : POKET,90 : G1=T : GOTO 1710
1710 T=G2 : GOSUB 3500 : FORR=1 TO Y : PRINT C$ : NEXT : GOSUB 3500 : PRINT "MUOVE (♦)" : P2$
1715 PRINT : PRINT "MOSSE " : GOSUB 2500 : POKET,63 : G2=T : GOTO 1700
2490 REM * SUBROUTINE PER INDIVIDUARE LE MOSSE POSSIBILI *
2500 M=0 : RESTORE

```

Listato «Isola» per VIC20

```

2510 FORJ=1T08
2520 READD
2530 IFPEEK(T+D)=102THENPRINTM$(J); M=1
2540 NEXT
2680 IFM=0THENGOSUB3500:GOTO3550
2690 REM * SUBROUTINE PER CONSENTIRE LA MOSSA EFFETTURATA *
2700 GETA$:IFA$=""THEN2700
2750 IFA$="5"THEN2900
2760 IFVAL(A$)<10RVAL(A$)>9THEN2900
2780 FORJ=1T09
2785 READD
2790 IFVAL(A$)=JANDPEEK(T+D)=102THENPOKET,32:T=T+D:RETURN
2800 NEXT
2810 REM * MESSAGGIO D' ERRORE *
2820 RESTORE:FORJ=1T08:READD:NEXT
2900 GOSUB3500
3000 FORR=1T010:PRINT"MOSSA NON VALIDA":FORR1=1T0200:NEXT:PRINT"J";C$:NEXT
3010 GOTO2700
3500 PRINT"S":FORR=1T016:PRINT:NEXT
3510 RETURN
3520 REM * SCRIVE IL VINCITORE *
3550 IFPEEK(T)=83THENV$=P1$
3560 IFPEEK(T)=90THENV$=P2$
3600 FORR=1T03:PRINTC$:NEXT
3610 GOSUB3500
3620 FORR=1T020:PRINTC$:PRINT" J W I N C E  "V$;" "
3624 FORR1=1T0100:NEXT:PRINT"J":NEXT
3630 PRINT"GIocate ANCORA (S/N)"
3640 GETN$:IFN$=""THEN3640
3650 IFN$="N"THENPRINT"ARRIVEDERCI !"
3660 IFN$="S"THENPRINT"J":GOTO270
3700 DATA42,44,46,-2,2,-46,-44,-42,42,44,46,-2,0,2,-46,-44,-42

```

Listato «Isola» per C64

```

10 REM *****
20 REM *
30 REM *      I S O L A
35 REM *
40 REM *      PERSONAL COMPUTER CLUB
45 REM *
50 REM *      COMMODORE 64
60 REM *
70 REM *****
80 REM
100 REM *** PROGRAMMA ISOLA DI ANDREA ACQUAVIVA ***
110 REM * VARIABILI STRINGA PER I SIMBOLI DELLE DIREZIONI *
120 M$(1)="  "M$(2)="  "M$(3)="  "M$(4)="  "
130 M$(5)="  "M$(6)="  "M$(7)="  "M$(8)="  "
140 C$=""
145 POKE53281,0
150 PRINT"J";" I S O L A  BY PERSONAL COMPUTER CLUB "
155 REM * ISTRUZIONI INIZIALI *
160 PRINT"PER MUOVERE USARE I TASTI INTORNO AL 5"
170 PRINT"  - 1 = "M$(1);"  - 2 = "M$(2);PRINT"  - 3 = "M$(3);
180 PRINT"  - 4 = "M$(4);PRINT"  - 6 = "M$(5);"  - 7 = "M$(6);

```


Listato Isola per C64

```

190 PRINT"00 - 8 = ";M$(7); " - 9 = ";M$(8):PRINT:PRINT
195 PRINT:PRINT" - PREMI UN TASTO PER CONTINUARE -"
200 GETR$:IFR$=""THEN200
210 REM * RICHIESTA NOMI E DIMENS. SCACCHIERA *
220 PRINT"0");"  I S O L A  BY PERSONAL COMPUTER CLUB "
230 PRINTTAB(10)"000NOME DEI GIOCATORI":PRINT
240 PRINT:INPUT" 1° GIOCATORE ";P1$
250 PRINT:INPUT" 2° GIOCATORE ";P2$
270 PRINT"00SCEGLI LE DIMENSIONI DELLA SCACCHIERA : "
280 PRINT"( MINIMO 7 X 5 , MASSIMO 9 X 7 )":PRINT
290 INPUT"ORIZZ. ";X
300 PRINT:INPUT"VERT. ";Y
990 REM * DISEGNO DELLA SCACCHIERA E DEI SIMBOLI *
1000 PRINT"0");"  I S O L A  BY PERSONAL COMPUTER CLUB "
1200 A=1154:B=(X*2-2):C=55426
1250 FORR=1TOY:FORI=0TOBSTEP2:POKER+I,102:POKEC+I,1
1350 NEXT
1400 A=A+80:C=C+80
1450 PRINT:PRINT
1500 NEXT
1550 G1=1394:G2=1312+I
1600 POKEG1,90:POKEG2,83
1640 GOSUB3500:PRINT"CHI GIOCA PER PRIMO ";
1650 INPUT" 1/ 0/ 2/ ";P$
1660 PRINT"0");C$
1670 P=VAL(P$)
1680 IFR<1ORP>2THEN1640
1690 IFR=2THEN1710
1695 REM PRINCIPALE *** 4 LINEE PER LA GESTIONE DEL GIOCO ***
1700 T=G1:GOSUB3500:PRINTC$;C$:PRINT"MUOVE: ( ♠ ) ";P1$;C$
1705 GOSUB3500:PRINT"MOSSA : ";GOSUB2500:POKET,90:G1=T:GOTO1710
1710 T=G2:GOSUB3500:PRINTC$;C$:PRINT"MUOVE: ( ♥ ) ";P2$;C$
1715 GOSUB3500:PRINT"MOSSA : ";GOSUB2500:POKET,83:G2=T:GOTO1700
2490 REM * SUBROUTINE PER INDIVIDUARE LE MOSSE POSSIBILI *
2500 M=0:RESTORE
2510 FORJ=1TO8
2520 READD
2530 IFFEEK(T+D)=102THENPRINTM$(J);M=M+1
2540 NEXT
2560 IFR=0THENGOSUB3500:GOTO3550
2590 REM * SUBROUTINE PER CONSENTIRE LA MOSSA EFFETTUATA *
2700 GETR$:IFR$=""THEN2700
2750 IFR$="5"THEN2900
2760 IFRVAL(R$)<10RVAL(R$)>9THEN2900
2780 FORJ=1TO9
2785 READD
2790 IFRVAL(R$)=JANDPEEK(T+D)=102THENPOKET,32:T=T+D:RETURN
2800 NEXT
2810 REM * MESSAGGIO D' ERRORE *
2820 RESTORE:FORJ=1TO8:READD:NEXT
2900 GOSUB3500
3000 FORP=1TO10:PRINT"MOSSA NON VALIDA" FORR=1TO200:NEXT:PRINT"0");C$;NEXT
3010 GOTO2700
3500 PRINT"0");FORR=1TO18:PRINT:NEXT
3510 RETURN
3520 REM * SCRIVE IL VINCITORE *

```

Listato Isola per C64

```

3550 IFPEEK(T)=83THENV$=P1$
3560 IFPEEK(T)=89THENV$=P2$
3570 FORR=1TO3:PRINTC$:NEXT
3580 PRINT"*" FORR=1TO17:PRINT NEXT
3590 FORR=1TO20:PRINTC$:PRINTTAB(5,"") V I N C E " (V$)"
3600 FORR1=1TO100:NEXT:PRINT"0":NEXT
3610 PRINT"NOGLOCATE ANDORA ( S / N )"
3620 GETN$:IFN$=""THEN3640
3630 IFN$="N"THENPRINT"ARRIVEDERCI !"
3640 IFN$="S"THENPRINT"0":GOTO270
3650 IPTA78,80,82,-2,2,-82,-80,-78,78,80,82,-2,0,2,-82,-80,-78

```

Listato Prugne

READY.

```

10 PRINT"
20 PRINT"
30 PRINT"
35 PRINT
40 PRINT"
50 PRINT"
60 PRINT
65 PRINT"
70 PRINT"
80 PRINT
90 PRINT
100 GOTO50000
200 PA=1064:SC=0:V=54272:C4=0:NS=0
210 C(1)=10:FORX=2TO10:C(X)=2:NEXTX:NF=3
220 W1=9:W2=5:W3=11:W4=11:W5=6:W6=3:FORX=1TO10:S(X)=0:NEXT
300 POKE53281,W6:GOSUB42000:GOSUB43000
1000 NS=NS+1
1100 REM
1102 REM POSIZIONE PRUGNE
1104 REM
1110 PF=1563:C1=0:C2=0:NG=0:FX=0:FY=-1:NT=0:FORX=1TO3:SG(X)=1:NEXT
1120 P(1)=1923:P(2)=1363:P(3)=1554:P(4)=1572:P(5)=1314:P(6)=1332
1140 P(7)=1158:P(8)=1168:P(9)=1758:P(10)=1768
1292 REM
1294 REM ESPLODO VIDEO ED ENTRO IN BATTAGLIA
1296 REM
1300 GOSUB40100
1350 GOSUB30000
1372 REM
1374 REM USCITA DALLA BATTAGLIA
1375 REM C9=0 NULLA DI DECISIVO - TORNO A BATTAGLIARE
1376 REM C9=1 DISTRUTTO UN FUNGO
1377 REM C9=2 DISTRUTTE TUTTE LE PRUGNE - SCHERMO SUCCESSIVO
1378 REM C9=3 DISTRUTTI TUTTI FUNGHI - FINE PARTITA
1379 REM
1380 IFC9=0THEN1350
1382 IFC9=2THENPRINTCHR$(31):PRINT"XXXXXXXXXXXX" OTTIMO "NS"!
1383 IFC9=2THENFORT=1TO1500:NEXT
1385 PRINT"X":PRINTCHR$(31);
1390 ONC9GOTO2100,1500,2000
1492 REM
1494 REM TUTTE PRUGNE K.O.
1496 REM
1500 EX=INT((RND(1)*100)+1)+50*NS:SC=SC+EX
1510 IFSC=PEANDC4=0THENC4=1:NF=Nf+1
1520 PRINT"XXXXX" SITUAZIONE
1530 PRINT"X" PUNTI "SC-EX
1540 PRINT"X" BONUS "EX
1550 PRINT"X" TOTALE "SC"
1560 PRINT"XXXXX" ('C' PER CONTINUARE)
1570 GOSUB44600
1580 GETB$:IFB$=""THEN1580
1590 IFB$<"C"THEN1580
1600 NP=NW:FORX=1TONW:S(X)=0:NEXT:GOSUB43000:GOTO1000
1992 REM
1994 REM TUTTI FUNGHI K.O.
1996 REM
2000 PRINT"XXXXX" GAME OVER
2010 PRINT"XX" COMPLIMENTII
2020 PRINT"XX" PUNTI "SC"
2030 PRINT"XXXXX" ('C' PER CONTINUARE)
2050 GETC$:IFC$=""THEN2050

```

Le «prugne» per il C.64

Vi proponiamo «Prune» un interessante programma realizzato da **Antonio Martino**. L'originalità del gioco ed una grafica, rendono questo programma molto interessante. Lasciamo la parola al nostro lettore per descrizione e commenti.

Considerazioni generali

«Prune» è stato ideato per un Commodore 64, ed è scritto interamente in Basic. È un gioco a tutto schermo. Ogni entità mobile presente nel gioco può sconfinare dal video e rientrare dalla parte opposta. Non si tratta di colpire certi bersagli sparando proiettili evitando la collisione diretta, ma si deve cercare, tranne alcune eccezioni, proprio la collisione. Lo scopo del gioco è di manovrare un **funghetto** marrone al fine di catturare delle prugne che corrono sullo schermo.

I comandi a disposizione del giocatore sono quelli per la direzione del proprio fungo, che si muove in 8 direzioni, secondo il presente schema:

U
Y I
G • K
B M
N

A disposizione anche i tasti «H» e «J», indifferentemente, per sganciare le **trappole** (cuoricini verdi).

Listato Le Prugne

```

30660 POKEHG,32:POKEHG+V,W6
30670 B(K)=G(K)+G(K)+40*GY(K)
30680 IFG(K)>2023THENG(K)=G(K)+1064-2024
30690 IFG(K)<1064THENG(K)=G(K)+2024-1064
30700 IFPEEK(G(K))=83THENM1=34:M2=175:GOSUB44200:GOSUB40700:GOTO30740
30710 IFPEEK(G(K))=90THENPOKEG(K),124:POKEG(K)+V,W3:GOTO31040
30720 IFPEEK(B(K))=32THENPOKEG(K),124:POKEG(K)+V,W3:GOTO30740
30730 GOTO30670
30740 NEXTK
30792 REM
30794 REM MOVIMENTO FUNGO
30796 REM
30800 D1=1
30805 GETB$
30810 IFB$="G"THENFX=-1:FY= 0:GOTO30890
30820 IFB$="Y"THENFX=-1:FY=-1:GOTO30890
30830 IFB$="U"THENFX= 0:FY=-1:GOTO30890
30840 IFB$="I"THENFX= 1:FY=-1:GOTO30890
30850 IFB$="K"THENFX= 1:FY= 0:GOTO30890
30860 IFB$="M"THENFX= 1:FY= 1:GOTO30890
30870 IFB$="N"THENFX= 0:FY= 1:GOTO30890
30880 IFB$="B"THENFX=-1:FY= 1:GOTO30890
30890 FF=PF:FU=PF+FX+40*FY
30900 IFFU>2023THENFU=FU+1064-2024
30910 IFFU<1064THENFU=FU+2024-1064
30912 POKEFF,32:POKEFF+V,W6
30914 IFB$<"H"ANDB$<"J"THEN30920
30915 IFNT=2*NTTHEN30920
30916 POKEFF,83:POKEFF+V,W2:NT=NT+1
30920 IFPEEK(FU)=83THENNT=NT-1:GOTO30970
30930 IFPEEK(FU)=87THENPOKEFU,90:POKEFU+V,W1:GOTO31040
30940 IFPEEK(FU)=124THENPOKEFU,90:POKEFU+V,W1:GOTO31040
30950 IFPEEK(FU)=91THENPOKEFU,90:POKEFU+V,W1:GOSUB41400:M1=34:M2=175:GOSUB44200
30960 IFC9=2THEN31100
30970 PF=FU:POKEPF,90:POKEPF+V,W1
30980 D1=D1+1:IFD1<=2THEN30805
31020 GOTO31100
31040 M1=17:M2=37:GOSUB44200:C9=1:NF=Nf-1:IFNF=0THENC9=3
31050 PRINTCHR$(31):PRINT"0000000000" PECCATO "N$!":FORT=1T01500:NEXT
31100 RETURN
40092 REM
40094 REM ESPLODO VIDEO DI PARTENZA
40096 REM
40100 GOSUB40500
40110 FORX=1TONW:IFS(X)=1THEN40140
40130 POKEP(X),81:POKEP(X)+V,C(X)
40140 NEXTX
40150 POKEPF,90:POKEPF+V,W1:FORX=1TONW:NC(X)=0:NEXT
40180 PP=11-NS:IFPP<3THENPP=3:RETURN
40192 REM
40194 REM ESTRAZIONE CASUALE MOVIMENTO PRUGNE
40196 REM
40200 A=INT(RND(1)*8)+1
40210 IFA=1THENDX(K)=-1:DY(K)= 0:RETURN
40220 IFA=2THENDX(K)=-1:DY(K)=-1:RETURN
40230 IFA=3THENDX(K)= 0:DY(K)=-1:RETURN
40240 IFA=4THENDX(K)= 1:DY(K)=-1:RETURN
40250 IFA=5THENDX(K)= 1:DY(K)= 0:RETURN
40260 IFA=5THENDX(K)= 1:DY(K)= 1:RETURN
40270 IFA=7THENDX(K)= 0:DY(K)= 1:RETURN
40280 IFA=8THENDX(K)=-1:DY(K)= 1:RETURN
40285 RETURN
40292 REM
40294 REM PRUGNA CADUTA IN TRAPPOLA
40296 REM
40300 POKENU,32:POKENU+V,W6:NT=NT-1
40310 IFK>1THENSC=SC+INT(100/NP)*NS
40320 IFK=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*3)+2)
40330 IFSC=PEANDC4=0THENC4=1:Nf=Nf+1
40340 NP=NP-1:S(K)=1:NC(K)=0:GOSUB40500
40360 IFNP=0THENC9=2
40370 RETURN
40492 REM
40494 REM TESTATA CON PUNTI
40496 REM
40500 PRINTCHR$(31):PRINT"0000000000"
40520 PRINT"0000000000" PRUNE SCHERMO "N$" PUNTI: "SC:RETURN
40692 REM
40694 REM PRUGNINO CADUTO IN TRAPPOLA
40696 REM
40700 POKEG(K),32:POKEG(K)+V,W6
40710 NT=NT-1:NG=NG-1:SG(K)=1:Q(K)=0:PG(K)=0
40720 SC=SC+INT((RND(1)*50)+10)*NS:GOSUB40500:RETURN
40992 REM
40994 REM ESPLOSIONE PRUGNINO
40996 REM
41000 C5=0:C7=1
41020 FORI=1T03
41040 FORH=G(K)-1+((I-2)*40)TOG(K)-1+((I-2)*40)+2
41060 IFH=G(K)THEN41110

```

Se la prugna incappa in un cuoricino verde, viene distrutta. Il punteggio assegnato aumenta col diminuire delle prugne presenti, e con l'aumentare degli schermi; per l'intrappolamento della superprugna il punteggio così calcolato viene casualmente raddoppiato, triplicato, o quadruplicato.

Se la prugna incappa nel fungo, viene catturata: punteggio calcolato nel modo già visto, ma aumentato perché abbiamo saputo dirigere il fungo nella direzione giusta.

Quando tutte le prugne vengono catturate viene assegnato un «bonus» che aumenta con l'aumentare degli schermi, e si passa allo schermo successivo.

I prugnini

I prugnini grigi sono perfidi in sommo grado e non possono essere distrutti dal fungo. Bisogna evitarli e sperare che cadano in un cuoricino verde, unico modo per catturarli.

Al momento del lancio da parte della prugna, il prugnino corre in direzione opposta a quella della propria prugna per un numero di movimenti casuale, che aumenta con l'aumentare degli schermi.

Se incontra un fungo, lo distrugge.

Se incontra un altro prugnino, una prugna, o gli occhioni blu lasciati dalle prugne, il prugnino non si scompone più di tanto ed esegue un balzo al di là dell'ostacolo, proseguendo, come se niente fosse, nella sua marcia alla ricerca del nostro povero funghetto.

Arrivato alla fine della sua corsa (quando cioè ha esaurito il numero dei movimenti assegnatogli casualmente), il prugnino si autodistrugge, esplodendo intorno a sé della «spazzatura» grigia; questa spazzatura distrugge tutto (anche i nostri cuoricini verdi), tranne, naturalmente,

entità amiche sue (prugne, ecc.)

L'unico modo per debellare questo essere furibondo è sperare che cada in un cuoricino verde prima di esplodere. Il punteggio che viene assegnato è casuale (come tutti i punteggi) ma comunque abbastanza alto e aumenta con l'aumentare degli schermi.

I funghi

Abbiamo a disposizione 3 funghi. Al conseguimento di un certo punteggio (che varia a seconda del numero delle prugne contro le quali stiamo giocando) ci viene assegnato un funghetto extra. Alla cattura dell'ultimo fungo a nostra disposizione, il gioco finisce.

Il numero massimo di cuoricini verdi che possiamo lasciare è pari al doppio del numero delle prugne contro le quali stiamo giocando. Se il fungo passa su un proprio cuoricino verde, lo distrugge senza altre conseguenze. Se il numero di cuoricini verdi diminuisce (perché ci siamo passati sopra col fungo o perché hanno fatto il loro dovere catturando prugne e prugnini), allora possiamo depositarne altri, senza però poter superare, ovviamente, il massimo consentito.

Onde facilitare l'azione del funghetto, per ogni movimento di prugne e prugnini, vengono eseguiti due movimenti del fungo.

Fasi della battaglia

Ogni fase logica della battaglia si compone, nell'ordine, dei seguenti movimenti.

- 1 - Movimento della superprugna
- 2 - Movimento delle altre prugne
- 3 - Movimento dei prugnini
- 4 - Movimento del fungo (ripetuto 2 volte)

Naturalmente, per ogni movimento vengono controllate tutte le possibili collisioni onde intraprendere le azioni

Listato Le Prugne

```
41062 Z=H
41064 IFZ>2023THENZ=Z+1064-2024
41066 IFZ<1064THENZ=Z+2024-1064
41070 IFPEEK(Z)=87ORPEEK(Z)=124ORPEEK(Z)=81THEN41110
41080 IFPEEK(Z)=83THENNT=NT-1
41090 IFPEEK(Z)=90THENC5=1
41100 POKEZ,102:POKEZ+V,W4
41105 IFCS=1THEN41200
41110 NEXTH:NEXTI
41130 POKEG(K)+V,W6:NG=NG-1:SG(K)=1:Q(K)=0:PG(K)=0
41140 FORI=1TO3
41150 FORH=G(K)-1+((I-2)*40)/G(K)-1+((I-2)*40)/3
41152 Z=H
41154 IFZ>2023THENZ=Z+1064-2024
41156 IFZ<1064THENZ=Z+2024-1064
41160 IFPEEK(Z)<102THEN41180
41170 POKEZ,32:POKEZ+V,W6
41180 NEXTH:NEXTI
41200 RETURN
41292 REM
41294 REM COLLISIONE PRUGNA/FUNGO
41296 REM
41300 IFK>1THENSC=SC+INT(100/NP)*NS+INT(100/NP)
41330 IFK=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*4)+2)+INT(100/NP)
41340 NP=NP-1:S(K)=1:NC(K)=0:IFNP=0THENC9=2
41360 IFSC=PEANDC4=0THENC4=1:NF=NF+1
41370 GOSUB40500:RETURN
41392 REM
41394 REM COLLISIONE FUNGO/PRUGNA
41396 REM
41400 FORT=1TONW:IFS(T)=1THEN41420
41410 IFFU=P(T)THEN41440
41420 NEXTT
41440 IFT>1THENSC=SC+INT(100/NP)*NS+INT(100/NP)
41450 IFT=1THENSC=SC+INT(100/NP)*NS*INT((RND(1)*4)+2)+INT(100/NP)
41460 NP=NP-1:S(T)=1:NC(T)=0
41470 IFNP=0THENC9=2
41480 IFSC=PEANDC4=0THENC4=1:NF=NF+1
41490 GOSUB40500:RETURN
41992 REM
41994 REM ISTRUZIONI PRINCIPALI
41996 REM
42000 GOSUB43200:PRINTCHR$(31)
42010 PRINT"PRUGNE":PRINT
42030 PRINT"SCOPO DEL GIOCO E' DI MANOVRARE UN"
42040 PRINT"FUNGO PER COLPIRE DELLE PRUGNE."
42050 PRINT"OLTRE ESSE C'E' UNA SUPERPRUGNA, PIU'"
42060 PRINT"CATTIVA MA DI MAGGIOR VALORE."
42070 PRINT"DEVI EVITARE I 'PRUGNINI' SPARATI DALLE"
42080 PRINT"PRUGNE. CON I TASTI 'H' O 'J' PUOI"
42090 PRINT"LASCIARE DELLE TRAPPOLE, E DEVI EVITARE"
42100 PRINT"DI CADERE IN QUELLE LASCIATE DALLE"
42110 PRINT"PRUGNE. FUNGO EXTRA A 'PE' PUNTI."
42120 PRINT"PUOI MUOVERTI IN 8 DIREZIONI,"
42130 PRINT"SECONDO IL SEGUENTE SCHEMA"
42140 PRINT
42150 PRINT"          U"
42160 PRINT"          Y I"
42170 PRINT"          G O K"
42180 PRINT"          B M"
42190 PRINT"          N"
42210 PRINT"(UN TASTO PER INIZIARE)";
42220 GETC$:IFCS=""THEN42220
42450 RETURN
42992 REM
42994 REM COUNTDOWN
42996 REM
43000 PZ=1355:PW=1636
43020 PRINTCHR$(31)
43030 PRINT"COUNTDOWN"
43040 PRINT"00000000 CREDIT:"
43060 FORM=1TONP
43080 POKEPZ-2+2*M,81:POKEPW-2+2*M+V,2
43090 NEXTM
43100 FORM=1TONF
43110 POKEPW-2+2*M,90:POKEPW-2+2*M+V,W1
43120 NEXTM
43130 FORZ=1346TOFZ+2*NF:OZ=0
43135 IFPEEK(Z)=81THEND2=1
43140 POKEZ-1,32:POKEZ-1+V,W6
43150 POKEZ,90:POKEZ+V,W1:IFD2=1THENGOSUB4000
43155 FORT=1TO100:NEXTT
43160 NEXTZ:PRINT"0"
43170 RETURN
43192 REM
43194 REM SCELTA NUMERO PRUGNE
43196 REM
43200 PRINTCHR$(31)
43210 PRINT"PRUGNE":PRINT"00000000"
43230 INPUT"CONTRO QUANTE PRUGNE VUOI GIOCARE";F$
```

Su sfondo nero, il titolo è scritto in blu con tre cornici: bianca, rossa e verde. A questo punto non ci resta che premere un tasto e partirà il nostro programma normalmente.

Lista delle variabili
listato «Inserire titolo»

AS - titolo da visualizzare e, alla fine, argomento del get.
T - posizione di partenza della cornice.

K - contatore della lunghezza della cornice.

**Prenota
Personal
Computer
in edicola**

Inserire il titolo sul C64

«**T**itolo» va inserito all'inizio di ogni pro-

gramma o utilizzato come subroutine.

Dato il run, compare il titolo del programma, già opportunamente inserito alla

-19 RUN 12

Speciale Commodore 57

Spiegazione delle istruzioni listato «Inserire titolo»

12. Pulisce lo schermo e imposta lo sfondo nero. Il chr\$ (142) imposta il set di caratteri standard (maiuscolo-grafico).

13. Al posto degli asterischi va inserito il titolo, che può avere una lunghezza qualsiasi (purché non sia più lungo della riga video)

14. Posiziona il cursore al centro del video, posizione che dipende dalla lunghezza del titolo.

15. Forma la parte superiore della cornice esterna di colore verde.

16. Forma la parte superiore della cornice centrale di colore bianco.

Attenzione! Prima della 'E' che imposta il bianco (CTRL+2), si ha il carattere che si ottiene con COMMODORE+M, così anche dopo la freccia in su (CTRL+6) si ha il simbolo ottenuto con CMDR+G.

17. Tra virgolette abbiamo: CTRL+6, CMDR+M, CTRL+2, SHIFT+B, CMEDR+3, SHIFT+0; Poi: SHIFT+P, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.

18. CMDR+M, CTRL+2, SHIFT+B, CMDR+3, CMDR+G, CMDR+7; Poi: CMDR+3, CMDR+M, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.

19. CTRL+6, CMDR+M, CTRL+2, SHIFT+B, CMDR+3, SHIFT+L; Poi: SHIFT+chiocciolina, CTRL+2, SHIFT+B, CTRL+6, CMDR+G.

20. CMDR+M, CTRL+2, ecc..., CTRL+6, CMDR+G.

21. Chiude la cornice più esterna.

22. Attende che venga premuto un tasto.

23. Da qui puoi inserire il tuo programma, che avrà ora un aspetto molto più 'professional', non trovi?

Naturalmente, chi vorrà avere il programma già registrato su cassetta, non dovrà fare altro che telefonarmi o scrivermi.

Incorniciare lo schermo

Presentiamo nel listato 1 una piccola utility per VIC 20, adattabile con poche modifiche al C 64, per stampare una cornice attorno allo schermo. È stata realizzata dal coordinatore del club di Personal Computer di Bra: Massimo Fumero.

Nel listato 2, Massimo mostra invece come stampare una frase in una certa posizione dello schermo, individuata da X e Y:

Il programma fa uso della funzione LEFT\$ (stringa, N), che permette di stampare i primi N caratteri della stringa partendo da sinistra.

In questo caso, la stringa è composta da 22 caratteri corrispondenti a «cursor down».

LEFT\$ («22 cursor down, 13) stampa i primi 13 caratteri della stringa partendo da sinistra, quindi 13 «cursor down»; il cursore verrà quindi posizionato sulla 13esima riga.

Questo piccolo truccetto vi permette di aggirare il problema della mancanza di un'istruzione VTAB.

Se il trucco dovesse essere usato più volte all'interno di un programma, consiglio di definire una stringa = «22 cursor down» all'inizio, di usare una variabile per la tabulazione verticale e una per quella orizzontale, e di richiamare il truccetto come una subroutine.

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆
☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

Auguri

Incorniciare lo schermo

Listato 1

```
10 GOSUB 500
250 END
500 REM *** CORNICE SCHERMO ***
505 POKE 36879,13 : PRINT "(CTRL-2)".
510 I = 7680 : J = 484 : S = 81 : PRINT "(SHIFT-CL)"
515 FOR K = I TO I + 21 : POKE K,S : NEXT
520 FOR K = I + 22 TO I + 22*21 STEP 22
525 POKE K,S : POKE K + 21,S : NEXT
530 FOR K = I + J TO I + J + 21 : POKE K,S : NEXT
535 PRINT "(HOME)"
540 RETURN
```

Listato 2

```
600 REM *** CURSORE ***
605 X = 13 : Y = 3
610 PRINT "(HOME)" LEFT$ ("22 cursor down", Y) TAB (X):
620 PRINT "PC - CLUB"
```

Commenti al listato 1

- Linea 505:** pone lo schermo nero, il bordo verde e il cursore bianco
- Linea 510:** variabili per adattare la subroutine al C 64
I = inizio locazioni di schermo
J = numero di caratteri dall'angolo alto sinistro a quello basso sinistro dello schermo
S = codice del carattere con cui verrà stampata la cornice, in questo caso il pallino (codice dec. 81)
- Linea 515:** stampa il margine superiore della cornice
- Linee 520/525:** stampa i bordi laterali della cornice
- Linea 530:** stampa il bordo inferiore della cornice

Commenti al listato 2

- Linea 605:** variabili di tabulazione. In questo caso la scritta PC - CLUB verrà stampata sulla 13esima colonna (X) e sulla 3a riga (Y)
- Linea 610:** porta il cursore all'angolo superiore destro e scende alla 3a riga

Un archivio «fonetico»

In molte applicazioni dei computer è necessario archiviare liste di nomi. Ad esempio clienti e fornitori per il commercio, passeggeri per le agenzie di viaggio, clienti per i professionisti e pazienti per gli ospedali. Questo è un programma per un direttorio telefonico ed illustra alcune delle tecniche utilizzabili in archivi di ricerca ed amministrativi.

Il programma è scritto per il Commodore Pet a 40 colonne, ma non utilizza caratteristiche Basic di macchine specifiche se non nel formatore di schermo, descritto nel testo. Il programma può funzionare in 8K ma con un direttorio limitato.

Uno degli inconvenienti incontrati nella ricerca in un archivio di nomi è quello di non conoscere con precisione il nome cercato. Un esempio, il nome «Clarke», può essere capito come «Clark», «Clerk» o «Clerke». È chiaro che quando si cerca in un indice è utile conoscere tutti i nomi che sono foneticamente simili.

Il codice Soundex è stato progettato per questo scopo: è compito del programma codificare un nome nel suo Soundex equivalente, nel seguente modo:

1 - la prima lettera del nome è anche la prima del codice

2 - le lettere seguenti sono rimpiazzate in questo modo: B,F,P e V - rimpiazzate con P

C,G,J,K,Q,S,X e Z - rimpiazzate con S

M e N - rimpiazzate con M

L e R - codificate senza modifiche

A,E,I,O,U,W,H e Y - non codificate

tutte le consonanti con pronuncia similare sono raggruppate

3 - una sequenza ininterrotta di lettere con lo stesso valore di codifica viene rimpiazzata

Figura 1

WATT	GURNEY
WITHE	GRIM
WAITE	GOREN
WHITE	GREEN
WYATT	
TUTTI	TUTTI
CODIFICATI	CODIFICATI
CON WTAA	CON GRMA

con un singolo codice, sebbene non sia inclusa la prima lettera del nome. Può essere inclusa cambiando la linea di lettura 2000

SDS=LEFT\$(NMS,1):LS=SDS:N=O

4 - il codice è forzatamente lungo quattro caratteri, e viene troncato aggiungendo alla destra la lettera A.

La procedura per codificare è nelle linee da 2000 a 2160. Nell'esempio di figura 1 i nomi foneticamente simili sono codificati in modo identico, però le regole della pronuncia inglese insegnano che questa non è la regola. «Belvoir» viene codificato BLPR, però può essere pronunciato «Bearer», che ha il codice BPRA. Così «Bough» è codificato BSAA

Le variabili del listato «Archivio fonetico»

NN - numero di record assegnati all'indice

TPS(NN) - indice telefonico

PT%(NN) anelli record

RCS - ingresso indice

NMS - nome

SXS - sesso

ADS - indirizzo

TLS - numero telefonico

SDS - codice Soundex

NP - puntatore per il successivo record

IP - puntatore del record precedente

CP - puntatore del record corrente

FU - puntatore del primo record dell'indice

FF - puntatore del primo record inutilizzato

IPS - chiave per azionare Input

LFS - nome del file da caricare

SFS - nome del file da salvare

UD - bandiera di aggiornamento, fissata su 1 se il file è creato o emendato

ER - bandiera di errore

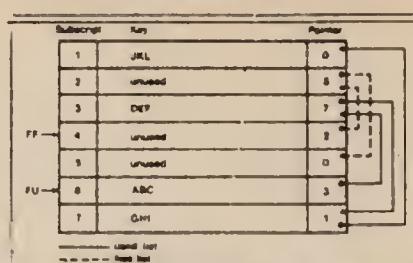
ND - numero di record da visualizzare

SES - carattere separatore, " "

CDS - stringa movimento cursore

AS, N, I, NS, LS, CS, J, K, L - variabili di uso comune.

Figura 2



ma viene anche pronunciato «Boff» o «Bow», che hanno il codice BPAA e BAAA.

Il codice Soundex è valido per cercare nomi in un piccolo direttorio telefonico personale, ma può solo formare una parte di un algoritmo di ricerca per un archivio più vasto. Generalmente le liste sono cercate per stadi successivi, con criteri di ricerca molto restrittivi e adatti ad ogni stadio. Così, il primo stadio può essere esaminato per un esatto riscontro sul cognome, iniziali, sesso e data di nascita. Se il riscontro richiesto è errato, il secondo stadio cerca sul Soundex sesso e anno di nascita. Il metodo di ricerca dipende dall'importanza assegnata alla ricerca esatta e alla richiesta di riscontri.

Varie strutture di dati posso-

no essere usate per creare un indice. La più semplice è una lista seriale o file. L'ultimo record della lista può essere seguito da nuove aggiunte, per cui la lista viene dichiarata grande a sufficienza per ulteriori espansioni. Le cancellazioni richiedono l'indice per cercare l'ingresso da togliere, e, queste, collocano nel record un valore speciale come Cancellato o «», la stringa nulla. Se vi sono molte cancellazioni, bisogna aggiungere una procedura per recuperare i record cancellati. L'indice deve esaminare ogni record per tutti i riscontri richiesti.

Una struttura molto pratica è la lista anulare. È usata nei programmi ed è largamente impiegata nelle strutture di dati comprensivi e nei sistemi di base dati commerciali. Ad ogni record è aggiunto un puntatore che indica la posizione del prossimo record in sequenza, il puntatore corrisponde al valore del numero del record appropriato. Dato che in Basic non è possibile mischiare variabili stringa e numeriche, bisogna inserire i puntatori in una lista separata, PT%, che è parallela alla linea dei dati principali, TPS.

All'inizio tutti gli elementi della lista sono assegnati a una lista libera. Il puntatore del primo elemento della lista è contenuto in FF e ogni elemento PT% è messo a puntare il seguente, in questo modo:

PT%(1)=2

PT%(2)=3

L'ultimo elemento PT% contiene zero, indicando il fine lista. Un altro puntatore, FU, dà il numero del primo record usato. La struttu-

Archivio fonetico - Procedure

- Caratteri di movimento cursore, particolari del Pet, sono visualizzati con codici interni a parentesi quadre. I codici usati sono: CD, cursore giù, HOM, cursore a casa e CLS, pulire schermo.

- **Procedura Master**, linee 200-310. Chiama l'inizio e le procedure di istruzione. Accetta l'ingresso dei tasti d'azione e chiama la procedura corrispondente.

- **Inizio**, linee 400-410. Fissa le costanti e pone UD=0, mostrando che all'inizio l'indice non richiede salvataggi. Open 1,0 in linea 410 apre la tastiera e un ingresso periferico. I dati possono essere accettati dalla tastiera con Input 1. Premendo il tasto Return si ha il vantaggio che Ready sul video è ignorato. Con Input 1 si ha lo svantaggio di non poter usare il promemoria e il Return che completa l'ingresso non produce una nuova linea sullo schermo. Questi svantaggi vengono facilmente eliminati e sono un piccolo prezzo da pagare in un programma che non visualizza Ready.

- **Istruzioni**, linee 600-665. Direttamente visualizzate sullo schermo.

- **Procedura di carico**, linee 800-890. Inizialmente NN è zero; se non lo è, un'indice è già stato creato/caricato. Il file nominato è aperto e letto e l'utente se lo desidera può estenderlo. Il record aggiunto viene inanellato nei record già esistenti dalle linee 860-870.

- **Procedura di controllo aggiunte**, linee 1000-1170. La linea 1000 assicura che non possono essere aggiunti record prima che il file sia stato caricato o creato. I dettagli del record da aggiungere sono nominati nelle linee 1010-1100, che controllano gli errori. Non si possono aggiungere stringhe con le virgole, in quanto nel Basic una virgola è un separatore. La subroutine 2000, richiamata dalla linea 1110, crea il codice Soundex. La linea 1120 crea il record concatenando le parti separate. La subroutine 2200, richiamata dalla linea 1130, aggiunge all'indice un nuovo record. Le linee 1440-1170 raggruppano la condizione di File Completo trovata dalla subroutine 2200. L'utente è invitato a salvare il file e a riavviare il programma ed espandere il file quando viene caricato.

- **Procedura di ricerca**, linee 1200-1290. Le linee 1200-1210 accettano la chiave di ricerca. Le linee 1220-1240 scorrono l'indice un record alla volta. La subroutine 2400 restituisce CP con il puntatore del successivo record sequenziale. La linea 1250 se non ci sono riscontri, visualizza un messaggio che riferisce il risultato negativo della ricerca. La linea 1270 stampa il record riscontrato e controlla che lo schermo non sia completo. Se lo è, le linee 1280-1290 permettono all'utente di esaminare i riscontri prima di visualizzare altri riscontri. Specifico del Pet è il Wait 59410,4,4 che aspetta sia premuta la barra spaziatrice. Se questa istruzione non è battuta correttamente il Pet sospende l'istruzione Wait bloccandone il ciclo.

Alcuni codici alternativi sono:

1290 GET AS:IF AS <> « » THEN 1290

1300 PRINT «[CLS]»:GOTO 1240

- **Salvataggio indice**, linee 1400-1430. L'indice viene salvato su un file, su cassetta di registrazione. Sul registratore, la virgola separatrice tra le variabili viene forzata includendola nelle istruzioni PRINT 2.

- **Nuovo indice**, linee 1600-1650. Sono richiesti i dettagli del nuovo file e viene fissata la lista corrispondente. Le linee 1640-1650 inanellano tutti gli elementi della lista nella lista inutilizzata.

- **Procedura di fine**, linee 1800-1850. Il file tastiera viene chiuso. Se l'indice è stato cambiato all'utente viene data l'opportunità di salvarlo.

- **Soundex**, linee 2000-2160. Vedere la descrizione nel testo.

- **Ingresso Nuove Aggiunte**, linee 2200-2350. Linea 2200, se FF=0 non vi sono record inutilizzati e il nuovo record non può essere aggiunto. La linea 2210 posiziona il record in TP\$(FF) e fissa un ciclo per trovare la locazione corretta in cui aggiungere il record; le linee 2220-2240 danno corpo al ciclo. Il ciclo quando si è trovato il punto corretto o quando tutti i record utilizzati sono stati esaminati e NP=0, li lascia entrambi. La linea 2250 aggiunge all'inizio dell'indice. Le linee 2260-2270 aggiungono all'indice il nuovo record. La linea 2280 aggiunge a un'indice vuoto. Le linee 2290-2300 aggiungono alla fine dell'indice.

- **Prossimo record**, linea 2400. Ripristina IP, CP e NP.

- **Visualizzazione dei record**, linee 2600-2690. La parte principale della procedura, linee 2610-2670, suddivide RC\$ in nome indirizzo e numero telefonico. L'attuale visualizzazione è alla linea 2680. La linea 2690 incrementa il contatore dei record visualizzati.

- **Cancellazione**, linee 2800-2910. La chiave del record da cancellare è immessa nella linea 2800, e l'indice analizzato per la ricerca nelle linee 2810-2840. Le linee 2850-2880 danno l'opportunità all'utente di rinunciare alla cancellazione. Le linee 2890-2910 cancellano il record come descritto nel testo.

- **Chiave d'ingresso**, linee 3000-3090. Il nome e il sesso sono accettati dalle linee 3020-3070, mentre il codice Soundex è creato dalla linea 3080.

ra è mostrata in figura 2.

Ogni nuovo record è assegnato alla posizione data da FF, il primo record della lista dei record inutilizzati; è il numero 7 nell'esempio. Il puntatore della lista è quindi portato al valore del puntatore corrispondente a questo record, nell'esempio 18.

Il nuovo record è incastrato nella lista dei record usati di modo che i record sono mantenuti in sequenza.

Quindi il puntatore del record della chiave GHK è fissato sul numero del nuovo record 7 e il puntatore di questo record è messo a puntare il record 4, precedentemente puntato da GHK.

Cancellare il record è più semplice. Il puntatore della lista libera FF è fissato sul numero del record da cancellare, mentre il puntatore di questo record è fissato sul valore precedente di FF, ponendo così il record cancellato alla testa della lista libera. Il record che prima puntava il record cancellato è puntato sul record precedentemente puntato dal record cancellato. Entrambe queste procedure richiedono codifiche aggiuntive per trattare con aggiunte e cancellazioni da e per l'inizio e la fine della lista. La ricerca è eseguita sequenzialmente seguendo i puntatori. Questo assicura che le chiavi siano analizzate in una sequenza ascendente e la ricerca è completata quando la chiave del record oltrepassa la chiave cercata.

La struttura a lista anulare è idonea ai record contenuti in RAM ma richiede l'aggiunta di uno o più indici quando si utilizza con il backing-store.

Può formare la base di un tale sistema, ed ha la virtù di permettere aggiunte e cancellazioni senza ristrutturare il file.

Il programma per eseguire queste procedure è stato preparato in maniera modulare, semplificando così la codifica, la prova e le successi-

ve espansioni. L'arte della codifica sta nel risolvere il problema e stabilirne la soluzione. Il primo passo è quello di definire il problema: la codifica può essere lasciata più tardi nella programmazione del ciclo. Se il programma è stato ben pianificato fin dal principio, codificare diventa, per la maggior parte, un processo meccanico. Il passo successivo è quello di ridurre la soluzione

a un fissaggio di moduli. A questo punto potete scegliere di usare uno pseudo linguaggio strutturato o una flowchart. La principale richiesta è che i moduli devono essere ben definiti, funzionali e sufficientemente piccoli, per essere chiari. È possibile codificare ogni uno in un ragionevole numero di linee.

Nel Basic, ogni modello si aspetta che il modulo con-

trollore possa essere codificato in una subroutine. Siccome è chiaro lo scopo dei moduli, e non possono essere troppo lunghi, non presentano difficoltà di codifica. Inoltre possono essere codificati in TOP-DOWN. Ogni subroutine è inizialmente codificata con una matrice, che può essere un semplice Return una linea per stampare il nome seguita da Return. Poi ogni subroutine è

codificata rimpiazzando la matrice, così il programma è gradualmente costruito. La matrice è necessaria dove può fissare valori in variabili per simulare la loro funzione, le procedure sono aggiunte e provate una per volta. Così il processo continua e le procedure più comuni possono essere usate frequentemente, cosicché possiate diventare più fiduciosi sulla loro correttezza.

Listato Archivio fonetico

Suoni similari

```

200 GOSUB 400
210 GOSUB 600
220 GET IPS:IF IPS="I" THEN 210
230 IF IPS="L" THEN GOSUB 800:GOTO 300
240 IF IPS="A" THEN GOSUB 1000:GOTO 300
250 IF IPS="F" THEN GOSUB 1200:GOTO 300
260 IF IPS="S" THEN GOSUB 1400:GOTO 300
270 IF IPS="N" THEN GOSUB 1600:GOTO 300
280 IF IPS="D" THEN GOSUB 2800:GOTO 300
290 IF IPS="E" THEN GOSUB 1800:END
295 GOTO 220
300 PRINT "  IMMETTERE IL CODICE PER LA SUCCESSIVA FUNZIONE,"
310 PRINT "OPPURE 'I' PER LE ISTRUZIONI":GOTO 220
400 SES="●":CDS="                ":UD=0
410 OPEN 1,0:RETURN
600 PRINT "  LE FUNZIONI DISPONIBILI SONO:-"
605 PRINT "  L - CARICARE UN'INDICE TELEFONICO"
610 PRINT "S - SALVARE UN'INDICE TELEFONICO"
615 PRINT "N - INSERIRE UN NUOVO INDICE TELEFONICO"
620 PRINT "A - AGGIUNGERE UN'INGRESSO ALL'INDICE"
625 PRINT "F - CERCARE L'INDICE"
630 PRINT "D - CANCELLARE UN'ENTRATA DELL'INDICE"
635 PRINT "E - FINE DEL PROGRAMMA"

```


Listato Archivio fonetico

```
640 PRINT "I - TORNA A QUESTO VIDEO"
645 PRINT " ORA, O IN RISPOSTA AL PROMEMORIA:-"
650 PRINT " IMMETTERE IL CODICE PER LA SUCCESSIVA FUNZIONE,"
655 PRINT "OPPURE 'I' PER LE ISTRUZIONI"
660 PRINT " IMMETTERE UNA DELLE LETTERE CITATE"
665 RETURN
800 IF NN<>0 THEN PRINT " UN INDICE E' GIA' STATO CARICATO":RETURN
805 PRINT " IMMETTERE IL NOME FILE":INPUT#1,LF$:PRINT
810 OPEN 2,1,0:INPUT#2,FU,FF,NN
820 PRINT " CI SONO";NN;"RECORD NEL FILE"
830 PRINT " QUANTI IN PIU'? ";:INPUT#1,AS:N=VAL(AS):PRINT
840 DIM TP$(NN+N),PT%(NN+N)
850 IF N=0 THEN 880
860 FOR I=NN+1 TO NN+N-1:PT%(I)=I+1:NEXT
870 PT%(NN+N)=FF:FF=NN+1
880 FOR I=1 TO NN:INPUT#2,PT%(I),TP$(I):NEXT
890 NN=NN+N:CLOSE 2:RETURN
1000 IF NN=0 THEN PRINT " USARE 'N' O 'L' PER CREARE O CARICARE
    UN FILE":RETURN
1010 PRINT " INSERIRE NOME, COGNOME. NON USARE":PRINT "VIRGOLE":PRINT
1020 INPUT#1,NM$:PRINT
1030 PRINT " INSERIRE SESSO('M' O 'F' O 'O' PER":PRINT "COMMERCIO
    ECC.)";
1040 INPUT#1,SX$:PRINT
1050 IF SX$="M" OR SX$="F" OR SX$="O" THEN 1070
1060 PRINT "'M' O 'F' O 'O' PER FAVORE. RIPETERE":GOTO 1040
1070 PRINT " INSERIRE INDIRIZZO. NON USARE VIRGOLE"
1080 PRINT:INPUT#1,AD$:PRINT
1090 PRINT " INSERIRE NUMERO TELEFONICO"
1100 PRINT:INPUT#1,TL$:PRINT
1110 GOSUB 2000
1120 RC$=SD$+NM$+SE$+AD$+SE$+TL$
```

Listato Archivio fonetico

```
1130 GOSUB 2200
1140 IF ER=0 THEN UD=1:RETURN
1150 PRINT " FILE PIENO. INSERIRE 'S' PER SALVARLO, QUINDI"
1155 PRINT " 'E' PER FINIRE. RIAVVIARE IL PROGRAMMA. USARE"
1160 PRINT " 'L' PER CARICARE IL FILE ED ESPANDERLO"
1165 PRINT " QUANDO AVETE CHIESTO DI VOLERE PIU'"
1170 PRINT " RECORD":RETURN
1200 GOSUB 3000:ND=0
1210 IF ER<>0 THEN RETURN
1220 GOSUB 2400
1230 IF SDE=LEFT$(TP$(CP),5) THEN 1270
1240 IF NP<>0 THEN 1220
1250 IF ND=0 THEN PRINT " NESSUN RISCONTRO":RETURN
1260 PRINT " TUTTI I RISCONTRI TROVATI":RETURN
1270 GOSUB 2600:IF ND-6*INT(ND/6)<>0 THEN 1240
1280 PRINT " BATTERE SPAZIO PER CONTINUARE"
1290 WAIT 59410,4,4:PRINT " ":GOTO 1240
1400 PRINT " INSERIRE IL NOME FILE":INPUT#1,SF$:PRINT
1410 OPEN 2,1,1:PRINT#2,FU;"",";FF;"",";NN
1420 FOR I=1 TO NN:PRINT#2,PT%(I);",";TP$(I):NEXT
1430 UD=0:CLOSE 2:RETURN
1600 IF NN<>0 THEN PRINT " AVETE GIA' UN FILE":RETURN
1604 PRINT " QUANTI RECORDS RICHIEDETE? ";
1605 INPUT#1,N$:NN=VAL(N$):PRINT
1610 IF NN<20 OR NN>250 THEN PRINT " 20-250 E' IL LIMITE.
      REINSERIRE";:GOTO 1605
1620 DIM TP$(NN),PT%(NN)
1630 FU=0:FF=1
1640 FOR I=1 TO NN-1:PT%(I)=I+1:NEXT
1650 PT%(NN)=0:RETURN
```


Listato Archivio fonetico

```
1800 IF UD<>1 THEN 1850
1810 PRINT " AVETE CAMBIATO IL FILE. VOLETE"
1820 PRINT "SALVARLO?"
1830 GET AS:IF AS="N" THEN 1850
1840 IF AS<>"Y" THEN 1830
1845 GOSUB 1400
1850 CLOSE 1:RETURN
2000 SDS=LEFT$(NMS,1):LS=" " :N=0
2010 FOR I=2 TO LEN(NMS):AS=MID$(NMS,I,1)
2020 IF AS="B" OR AS="F" OR AS="P" OR AS="V" THEN CS="P":GOTO 2090
2030 IF AS="C" OR AS="G" OR AS="J" OR AS="K" THEN CS="S":GOTO 2090
2035 IF AS="T" OR AS="D" THEN CS="T":GOTO 2090
2040 IF AS="Q" OR AS="S" OR AS="X" OR AS="Z" THEN CS="S":GOTO 2090
2050 IF AS="M" OR AS="N" THEN CS="M":GOTO 2090
2060 IF AS="L" OR AS="R" THEN CS=AS:GOTO 2090
2070 IF AS=" " THEN I=99:GOTO 2110
2080 LS=AS:GOTO 2110
2090 IF CS=LS THEN 2110
2100 SDS=SDS+CS:N=N+1:LS=CS
2110 NEXT
2120 IF N=3 THEN 2150
2130 IF N<3 THEN SDS=SDS+LEFT$( "AAA",3-N):GOTO 2150
2140 SDS=LEFT$(SDS,4)
2150 SDS=SDS+SXS
2160 RETURN
2200 ER=0:IF FF=0 THEN R=1:RETURN
2210 NP=FU:IP=0:CP=0:TP$(FF)=RCS
2220 IF NP=0 THEN 2280
2230 GOSUB 2400:AS=LEFT$(TP$(CP),5)
2240 IF SDS>AS THEN 2220
2250 IF IP=0 THEN FU=FF:GOTO 2270
```

Listato Archivio fonetico

```

2260 PT%(IP)=FF
2270 N=PT%(FF):PT%(FF)=CP:FF=N:GOTO 2350
2280 IF CP=0 THEN FU=FF:GOTO 2300
2290 PT%(CP)=FF
2300 N=PT%(FF):PT%(FF)=NP:FF=N:GOTO 2350
2350 RETURN
2400 IP=CP:CP=NP:NP=PT%(CP):RETURN
2600 RCS=TP$(CP):L=LEN(RCS):J=0:K=0
2610 FOR I=6 TO L
2620 IF MID$(RCS,I,1)<>"●" THEN 2650
2630 IF J=0 THEN J=I:GOTO 2650
2640 K=I
2650 NEXT
2660 NMS=MID$(RCS,6,J-6):ADS=MID$(RCS,J+1,K-J-1)
2670 TLS=RIGHT$(RCS,L-K)
2680 PRINT " ";NMS;TAB(20);TLS:PRINT ADS
2690 ND=ND+1:RETURN
2800 GOSUB 3000:IF ER<>0 THEN RETURN
2810 GOSUB 2400
2820 IF SDS=LEFT$(TP$(CP),5) THEN 2850
2830 IF NP<>0 AND SDS=LEFT$(TP$(CP),5) THEN 2810
2840 PRINT " PIU' NESSUN RISCONTRO":RETURN
2850 PRINT "      ":GOSUB 2600
2860 PRINT "  CANCELIA TE QUESTO 'RECORD? ('S' O 'N') "
2870 GET AS:IF AS="N" THEN 2830
2880 IF AS<>"S" THEN 2870
2890 IF IP=0 THEN FU=PT%(CP):GOTO 2910
2900 PT%(IP)=PT%(CP)
2910 PT%(CP)=FF:FF=CP:TP$(CP)=" ":UD=1:RETURN
3000 ER=0:IF NN=0 THEN PRINT " USATE 'N' O 'L' PER CREARE/CARICARE
    UN FILE":ER=1:RETURN

```


Listato Archivio fonetico

```

3010 IF FU=0 THEN PRINT " NESSUN RECORD SUL FILE":ER=1:RETURN
3020 PRINT " INSERITE NOME E COGNOME. NON USATE":PRINT "VIRGOLE":
      PRINT
3030 INPUT#1,NM$:PRINT
3040 PRINT " INSERIRE SESSO ('M' O 'F' O 'O' PER":PRINT "COMMERCIO
      ECC.)";
3050 INPUT#1,SX$:PRINT
3060 IF SX$="M" OR SX$="F" OR SX$="O" THEN 3080
3070 PRINT " 'M' O 'F' O 'O' PER FAVORE. RIPETERE":GOTO 3050
3080 GOSUB 2000:IP=0:CP=0:NP=FU
3090 PRINT " ":RETURN

```

I caratteri suonano sul VIC 20

Questa piccola routine
proposta da **Bruno**

Montresor, di Verona gira sul Commodore VIC 20 e permette la visualizzazione dei caratteri accompagnati da un suono. È un effetto insolito che può essere utile in molti casi (per esempio nella

presentazione di un gioco).

Il contenuto della variabile A\$ a riga 30-40 può essere cambiato a piacere e deve contenere il messaggio da visualizzare.

Auguri

Listato I caratteri suonano

```

1 REM ** CARATTERI SONORI **
2 REM ** PER VIC 20 **
10 POKE36879,8:PRINTCHR$(5)
20 PRINT"?"
29 REM ** INIZIO ROUTINE **
30 A$="QUESTA ROUTINE STAMPA SU SCHERMO CARATTERI E SUONI. "
40 A$=A$+"SI POSSONO AGGIUNGERE MESSAGGI A VOLONTA'. BUON LAVORO! "
100 FORQ=1TOLEN(A$):S$=MID$(A$,Q,1):PRINTS$;
110 POKE36875,200:POKE36878,15
120 FORI=1TO15:PRINT"■ ■";NEXTI
130 POKE36875,0:POKE36878,0
140 NEXTQ
141 REM ** FINE ROUTINE **

READY.

```

OLTRE LE BARRIERE DELL'HARD E DEL SOFT.

SVITM SPECTRAVIDEO

il computer del grande standard MSX



I Re dei Rack



Mobile per computer Mod. 84C1 - disponibile nei colori: noce, nero, bianco - Lit. 140.000 + IVA

MOBILI

Prandini

scrivete a:

PRANDINI MOBILI - Via Dante, 30 - Tel. 0425/81666
45030 CASTELNOVO BARIANO (RO)

Vi invieremo gratis il nostro Catalogo Generale a colori